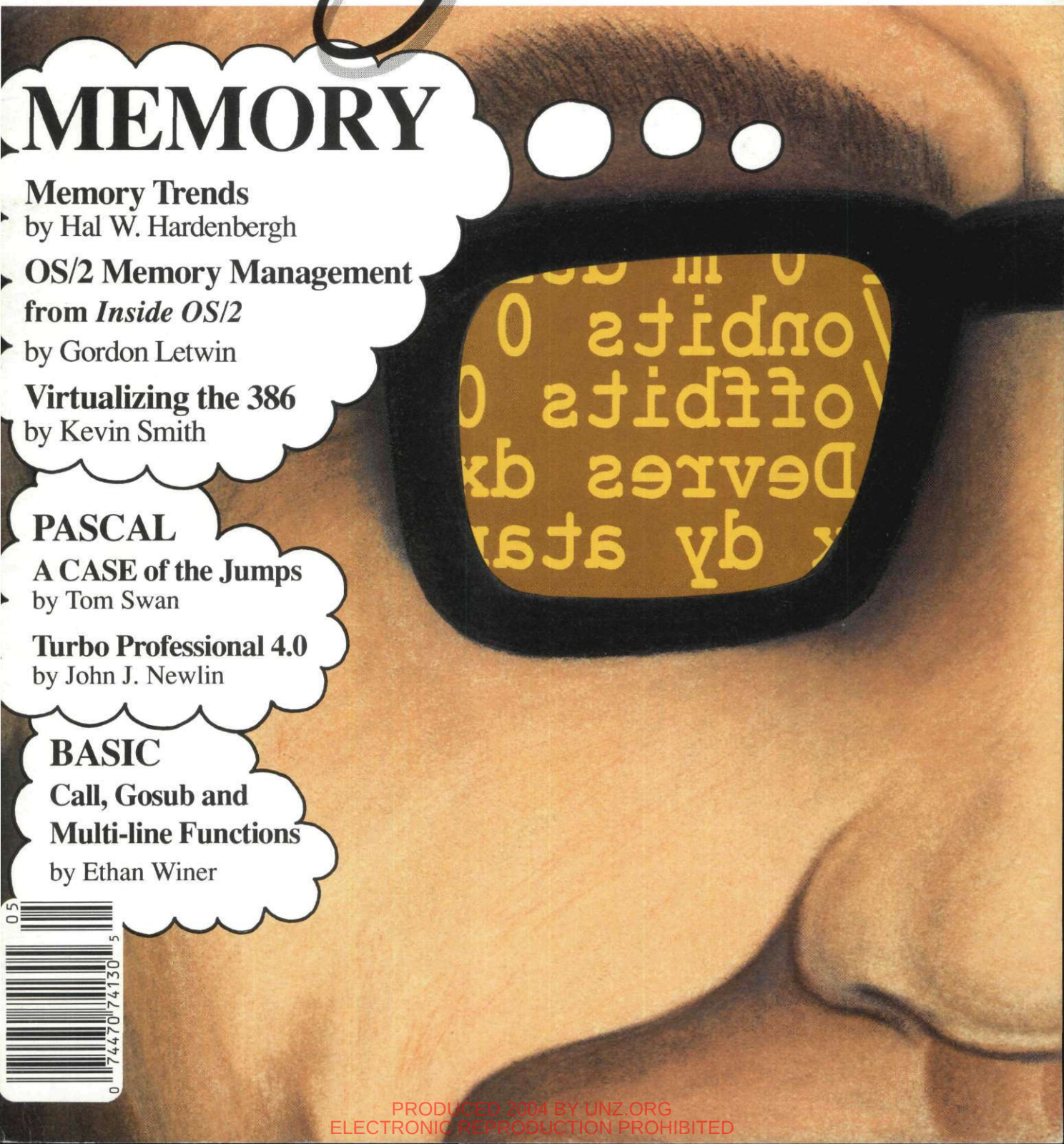
The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal

MAY/JUNE 1988
Volume 6.3
$5.00 Canada $6.00



MEMORY

Memory Trends
by Hal W. Hardenbergh

OS/2 Memory Management from *Inside OS/2*
by Gordon Letwin

Virtualizing the 386
by Kevin Smith

PASCAL

A CASE of the Jumps
by Tom Swan

Turbo Professional 4.0
by John J. Newlin

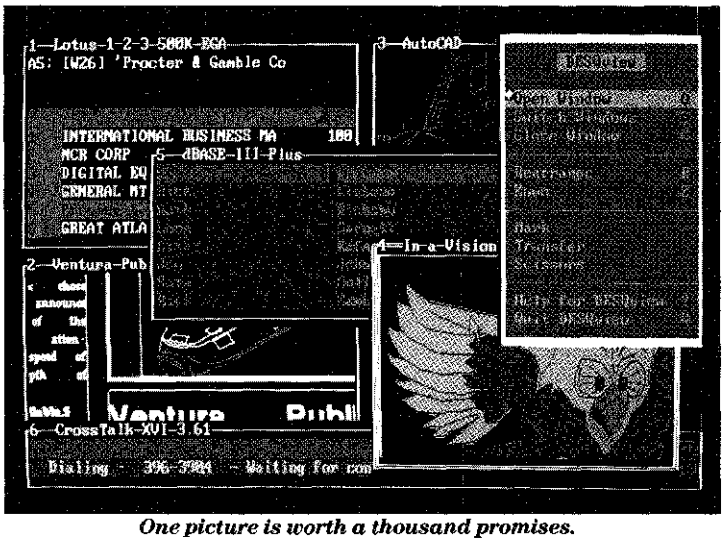
BASIC

Call, Gosub and Multi-line Functions
by Ethan Winer

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

How to tell the difference between DESQview™ 2.0 and any other environment.

Selecting DESQview, the environment of choice, can give you the productivity and power you crave, without the loss of your old programs and hardware. If you like your existing programs, want to use them together, transfer data between them, print, sort, communicate with or process-in-background, yet still have the need to keep in place your favorite PC(8088, 8086, 80286 or 80386), DESQview is the "proven true" multitasking, multi-windowing environment for you. Best of all, DESQview 2.0 is here now, with all the money saving, time saving, and productivity features that others can only promise for the all-too-distant future.



One picture is worth a thousand promises.

And with DESQview's new graphics enhancements for Hercules, CGA, EGA, and VGA, Version 2.0 still offers the same award winning and pioneering features for programs that earned DESQview its leadership, only now you can also run desktop publishing programs, CAD programs, even GEM-™, Topview-™, and Microsoft Windows-™ specific programs. In some cases you'll add as little as 10–40K to your system overhead. Now you can have multi-tasking, multi-windowing, break the 640K habit too and still get an auto dialer, macros, menus for DOS and, for advanced users, a new complete application programmer's interface capability. No wonder that over the years, and especially in recent months, DESQview, and now DESQview 2.0 have earned extravagant praise from some of the most respected magazines in the industry.

"Product of the Year" by readers vote in InfoWorld.

"Best PC Environment" by popular vote at Comdex Fall in PC Tech Journal's "System Builder" Contest.

"—I wouldn't want to run an IBM or compatible computer without DESQview"—InfoWorld, Michael Miller.

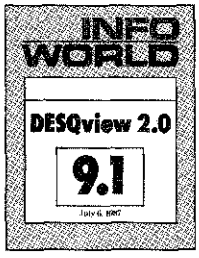
"A colossus among windowing environments"... "will run almost anything"—PC Week, Marvin Bryan.

"Windows, promises, but DESQview delivers"—MICROTIMES, Birrell Walsh.

No other environment has consistently pioneered features, openness, and productivity. See for yourself. Send in the coupon. The possibilities are endless with DESQview 2.0.

Attention Programmers: For more information about Quarterdeck's API, and future 386 program extensions, call us today.

SYSTEM REQUIREMENTS
IBM Personal Computer and 100% compatibles (with 8086, 8088, 80286 or 80386 processors) with monochrome or color display; IBM Personal System/2 • Memory: 640K recommended; for DESQview itself 0-145K • Expanded Memory (Optional): expanded memory boards compatible with the Intel AboveBoard; enhanced expanded memory boards compatible with the AST RAMpage • Disk: Two diskette drives or one diskette drive and a hard disk • Graphics Card (Optional): Hercules, IBM Color/Graphics (CGA), IBM Enhanced Graphics (EGA), IBM Personal System/2 Advanced Graphics (VGA) • Mouse (Optional): Mouse Systems, Microsoft and compatibles • Modem for Auto-Dialer (Optional): Hayes or Compatible • Operating System: PC-DOS 2.0-3.3; MS-DOS2.0-3.2 • Software: Most PC-DOS and MS-DOS application programs; programs specific to TopView 1.1, GEM 1.1 and Microsoft Windows 1.03 • Media: DESQview 2.0 is available on either 5¼" or 3½" floppy diskettes



Rush me DESQview 2.0! Today! PGMR-5/6

No. of Copies	Media 3½"/5¼"	Product	Retail Price ea.	Total
		DESQview 2.0	$129.95	$
		Shipping & Handling USA	$ 5.00	
		Outside USA	$ 10.00	$
		Sales Tax (CA residents)	6.5%	$
			Amount Enclosed	$

Payment: ☐ Visa ☐ MC ☐ AMEX ☐ Check

Credit Card: Valid Since ____/____ Expiration ____/____

Card Number:

Credit Card Name

Shipping Address

City State Zip Telephone

Mail to: Quarterdeck Office Systems, 150 Pico Boulevard, Santa Monica, CA 90405.
NOTE: If you own DESQview call us for a special upgrade offer, or send in your DESQview registration card. AST Special Edition users included.

DESQview™ Quarterdeck

Quarterdeck Office Systems • 150 Pico Boulevard, Santa Monica, CA 90405 • (213) 392-9851

DESQview is a trademark of Quarterdeck Office Systems. AboveBoard is a trademark of Intel Corporation. Hayes is a trademark of Hayes MicroComputer Products Inc. IBM, PC, Personal System/2 and TopView are trademarks of International Business Machines Corporation. Microsoft Windows and MS are registered trademarks of Microsoft Corporation. Mouse Systems is a trademark of Metagraphics/Mouse Systems. RAMpage is a trademark of AST Research, Inc. GEM is a trademark of Digital Research. Hercules is a trademark of Hercules.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

VEDIT PLUS

#1 PROGRAMMABLE EDITOR

Call 1-800-45-VEDIT for FREE Fully Functional Demo Disk

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS easily satisfies the most demanding computer professional.

Try a Dazzling Demo Yourself.

The free demo disk is fully functional—you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial—you experiment in one window while another gives instructions.

The powerful "macro" programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a "macro"—it shows that no other editor's "macro" language even comes close. And VEDIT PLUS is only 40K in size.

Go ahead. Call for your free demo today, You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Only VEDIT PLUS is this Flexible.

The installation lets you pick from closely emulating the keyboard layout of Word Perfect, WordStar and others. Or you can easily create your own layout and even your own editing functions. Supports any screen size—you pick screen colors and attributes.

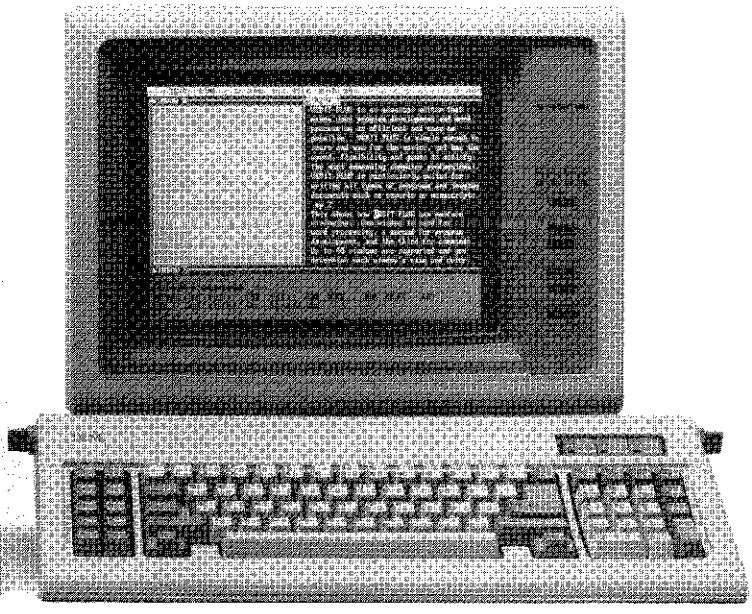
Supports the IBM PC, XT, AT and PS/2. Also supports MultiLink, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, FlexOS (protected mode), CP/M-86 and CP/M. (Yes, we support windows on most CRT terminals, including CRTs connected to an IBM PC.) Order direct or from your dealer. $185.

Special: VEDIT (single file, no windows) for CP/M—$49.

- **Fully Network Compatible**
- **Call for XENIX-286 version**
- **30 Day Money-back guarantee**

Compare Features and Speed

	BRIEF	Norton Editor	PMATE	VEDIT PLUS
'Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	Unlimited
Multiple file editing	20+	2	No	20+
Windows	20+	2	No	20+
Macro execution window	No	No	No	Yes
Pop-up menus	No	No	No	Yes
Execute DOS commands	Yes	Yes	Yes	Yes
Automatic processing of Compiler errors	Yes	No	No	Yes
"Cut and paste" buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
Convert to/from WordStar	No	No	No	Yes
On-line calculator	No	No	No	Yes
Configurable Keyboard	Hard	No	Hard	Easy
43 line EGA support	Yes	No	No	Yes
Manual size/index	250/No	42/no	469/Yes	380/Yes
Benchmarks in 120K File:				
2000 replacements	1:15 min	34 sec	1:07 min	6 sec
Pattern matching search	20 sec	Cannot	Cannot	2 sec
Pattern matching replace	2:40 min	Cannot	Cannot	11 sec



VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc. MultiLink and PC-MOS/386 are trademarks of the The Software Link, Inc. CP/M and FlexOS are trademarks of Digital Research. MS-DOS is a trademark of Microsoft.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, Wyse WY700 and others.
*Demo disk is fully functional, but does not readily write large files.

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299, TELEX 701821

Number **56** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers
PROGRAMMER'S Journal
Volume 6.3
May/June 1988

Contents

4 Notes from the Editors

9 Letters

16 PJ Forum: Memory Trends *by Hal W. Hardenbergh*
Can memory bus-bandwidth keep up with CPU performance?

20 Redirections: Pink Noise *by Frank D. Greco*
Technogossip from what used to be Leftfield

30 Consulting: Getting Your Consulting Tools in Order *by Paul Barkley*
You've taken care of your clients—now, take care of yourself

38 OS: OS/2 Memory Management excerpted from *Inside OS/2 by Gordon Letwin*
Microsoft's Chief Architect for Systems Software tells us how

46 386: Virtualizing the 386 *by Kevin Smith*
An argument for developing a 386 hypervisor and the nuts and bolts

56 Pascal: A CASE of the Jumps *by Tom Swan*
Add fast jump table CASE statements to Turbo Pascal 4.0

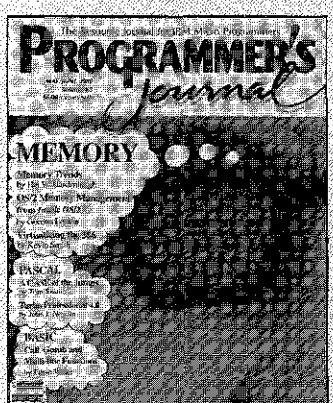
60 Pascal: Turbo Professional 4.0 *by John J. Newlin*
Compiled units, demonstration programs, and source code that rewards study

68 BASIC: Call, Gosub, and Multi-line Functions *by Ethan Winer*
Which approach you should favor—when

71 Miscellaneous *by M. Steven Baker*
80286/80386 stuff, graphics/windowing, parser generators, assemblers/tools, *Programmer's Guide to PC and PC/2 Video Systems*, *Advanced C Programming for Displays*

79 PJ Billboard

80 Advertiser Index



Having enough memory is only the beginning . . . managing it is another matter altogether. Our house programmer, P. J. Listings, shares his thoughts on this issue's contents. Cover art by Ralph Bentley

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Interlocking Pieces: Blaise and Turbo Pascal.

Now, for Turbo Pascal 4.0!

Whether you're a Turbo Pascal expert or a novice, you can benefit from using professional tools to enhance your programs. With Turbo POWER TOOLS PLUS™ and Turbo ASYNCH PLUS,™ Blaise Computing offers you all the right pieces to solve your 4.0 development puzzle.

Compiled units (TPU files) are provided so each package is ready to use with Turbo Pascal 4.0. Both POWER TOOLS PLUS and ASYNCH PLUS use units in a clear, consistent and effective way. If you are familiar with units, you will appreciate the organization. If you are just getting started, you will find the approach an illustration of how to construct and use units.

◆POWER TOOLS PLUS is a library of over 180 powerful functions and procedures like fast direct video access, general screen handling including multiple monitors, VGA and EGA 50-line and 43-line text mode, and full keyboard support, including the 101/102-key keyboard. Stackable and removable windows with optional borders, titles and cursor memory provide complete windowing capabilities. Horizontal, vertical, grid and Lotus-style menus can be easily incorporated into your programs using the menu management routines. You can create the same kind of moving pull down menus that Turbo Pascal 4.0 uses.

Control DOS memory allocation. Alter the Turbo Pascal heap size when your program executes. Execute any program from within your program and POWER TOOLS PLUS automatically compresses your heap memory if necessary. You can even force the output of the program into a window!

Write general interrupt service routines for either hardware or software interrupts. Blaise Computing's unique intervention code lets you develop memory resident (TSRs) applications that take full advantage of DOS capabilities. With simple procedure calls, "schedule" a Turbo Pascal procedure to execute either when pressing a "hot key" or at a specified time.

◆ASYNCH PLUS provides the crucial core of hardware interrupts needed to support asynchronous data communications. This package offers simultaneous buffered input and output to both COM ports, and up to four ports on PS/2 systems. Speeds to 19.2K baud, XON/XOFF protocol, hardware handshaking, XMODEM (with CRC) file transfer and modem control are all supported. ASYNCH PLUS provides text file device drivers so you can use standard "Readln" and "Writeln" calls and still exploit interrupt-driven communication.

The underlying functions of ASYNCH PLUS are carefully crafted in assembler and drive the hardware directly. Link these functions directly to your application or install them as memory resident.

Blaise Computing products include all source code that is efficiently crafted, readable and easy to modify. Accompanying each package is an indexed manual describing each procedure and function in detail with example code fragments. Many complete examples and useful utilities are included on the diskettes. The documentation, examples and source code reflect the attention to detail and commitment to technical support that have distinguished Blaise Computing over the years.

Designed explicitly for Turbo Pascal 4.0, Turbo POWER TOOLS PLUS and Turbo ASYNCH PLUS provide reliable, fast, professional routines – the right combination of pieces to put your Turbo Pascal puzzle together. **Complete price is $129.00 each.**

THE BLAISE MENU

Turbo POWER SCREEN *$129.00*
NEW! General screen management: paint screens; block mode data entry or field-by-field control with instant screen access. Now for Turbo Pascal 4.0, soon for C and BASIC.

Turbo C TOOLS *$129.00*
Full spectrum of general service utility functions including: windows; menus; memory resident applications; interrupt service routines; intervention code; and direct video access for fast screen handling. For Turbo C.

C TOOLS PLUS *$129.00*
Windows; menus; ISRs; intervention code; screen handling and EGA 43-line text mode support; direct screen access; DOS file handling and more. Specifically designed for Microsoft C 5.0 and QuickC.

ASYNCH MANAGER *$175.00*
Full featured interrupt driven support for the COM ports; I/O buffers up to 64K; XON, XOFF; up to 9600 baud; modem control and XMODEM file transfer. For Microsoft C and Turbo C or MS Pascal.

PASCAL TOOLS/TOOLS 2 *$175.00*
Expanded string and screen handling; graphics routines; memory management; general program control; DOS file support and more. For MS Pascal.

KeyPilot *$49.95*
"Super-batch" program. Create batch files which can invoke programs and provide input to them; run any program unattended; create demonstration programs; analyze keyboard usage.

EXEC *$95.00*
NEW VERSION! Program chaining executive. Chain one program from another in different languages; specify common data areas; less than 2K of overhead.

RUNOFF *$49.95*
Text formatter for all programmers. Written in Turbo Pascal; flexible printer control; user-defined variables, index generation and a general macro facility.

TO ORDER CALL TOLL FREE
800-333-2057

TELEX NUMBER - 338139

YES! **Send me the right pieces!**
Enclosed is $_____ for _____ copies of _____.
☐ Please send me more information on your products.
CA residents add Sales Tax. Domestic orders add $4.00 for UPS shipping, $10.00 for Federal Express standard air.
Name: _____ Phone: (_____) _____
Address: _____
City: _____ State: _____ Zip: _____
VISA or MC#: _____ Exp. Date: _____

Microsoft and QuickC are registered trademarks of Microsoft Corporation. Turbo Pascal is a registered trademark of Borland International.

◆

BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

Number 2 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers
PROGRAMMER'S Journal

Editor
Erin O'Connor
Technical Editor
M. Steven Baker
Contributing Editors
Michael Abrash, Frank D. Greco, Tom Swan

Contributing Writers
M. Steven Baker
Paul Barkley
Frank D. Greco
Hal W. Hardenbergh
Gordon Letwin
John J. Newlin
Kevin Smith
Tom Swan
Ethan Winer

Publisher
Elizabeth Oakley
Production
Ralph Bentley
Illustration
Jim Carpenter
Advertising
Robert Tappan
Subscriptions
Gary Hoyer
Accounting
Lori Fanslow
Business Manager
Bobbi Sinyard
Founder
Greg Estes

Business and Editorial Offices
Programmer's Journal
150 North 4th Street
Springfield, OR 97477
(503) 747-0800

Subscriptions: 1 year (6 issues) $24
($30 in Canada and $40 elsewhere)
Change of Address
Programmer's Journal
P.O. Box 30160
Eugene, OR 97403

Articles and letters in Programmer's Journal represent the views of their authors and not necessarily those of the publisher. Letters to the editor become the property of Programmer's Journal and may be edited and published.

Code and listings in Programmer's Journal appear "as is," and Programmer's Journal makes no guarantee or warranty of their quality, performance, accuracy, or suitability for any particular purpose. Programmer's Journal assumes no responsibility or liability to any person or entity with respect to any damage caused or alleged to be caused by their use.

Advertising: For ad rates, deadlines, and placement, call Robert Tappan (503/747-0800), or write to Programmer's Journal, P.O. Box 30160, Eugene, OR 97403.

Programmer's Journal is produced on Ventura Publisher, a product of Xerox Corporation.

Programmer's Journal (ISSN 0747-586) is published bi-monthly by Oakley Publishing Company. Second-class postage paid at Springfield, OR, and additional mailing offices.

© 1988 OAKLEY PUBLISHING COMPANY

NOTES FROM THE EDITORS

We're pleased to announce that Michael Abrash has consented to join Contributing Editors Frank Greco and Tom Swan on the *PJ* masthead. Michael's first act on the new job was to give Michael Abrash a vacation. Michael will continue his series on programming for the VGA in our next issue.

*

One of the best ad campaigns we've ever seen was addressed to potential *Computer Language* advertisers. Carl Landau's "Buy an ad. Get a cat." had earnest members of the ASPCA in an uproar, but most of us loved the inanity of it all. You had to see the photo of Carl's cat to get the full effect. Robin Tappan has suggested that we approach our own potential advertisers with "Buy an ad. Get an article," a reflection of the direct relationship between ad revenues and magazine size. If you like this magazine, say so when you talk to advertisers—even if you use bingo cards from *BYTE* or *Dr. Dobb's*. We have so many good articles to run and not enough editorial space. One consequence is that our writers are getting a reputation for dense prose. We'd love to have the space for more leisurely exposition, but until we can sell more ads, you'll have to struggle with complex sentences.

*

"OS/2 is almost in its beta test phase—we will wait six to nine months for a little maturation. Bill Gates is *late!*" So reads one of the more expansive responses to our question about developing for OS/2 on last issue's Peer Review card. Most of you contented yourselves with a simple "no."

We're late, too. Our coverage of OS/2 has been pretty light or of the bashing variety, so we're taking a shortcut—right to the source. Gordon Letwin is the architect of OS/2 and in this issue lays out the rationale and details of how OS/2 manages memory. We're not out to convert you, but we do feel obliged to inform your decisions about the operating environments you'll choose to develop for.

Steve Baker complains below, and so have several of you, about late products for development whose ultimate configurations grow out of political rather than the most appropriate engineering decisions. In the latter half of the 80s we seem fated to work around, make do, and wish it had been done differently. Not content with kludges and wishful thinking, Kevin Smith makes a concrete proposal for alterations to the 386 in this issue. Don't throw that old operating system away!—*Erin*

In February, I went down to the Software Development '88 Conference, a series of workshops put on by *Computer Language*, *AI Expert*, and the *Unix Review*. This well-organized conference was their first get-together not devoted exclusively to C, and the workshops covered a diversity of subjects presented by some excellent speakers. I may never program in Lisp, Ada, or an object oriented database, but at least I have a better feel for these languages. At *PJ*, we try to cover the leading edge of technology along with our traditional beat. Personally, I'm a pragmatic programmer. I'm intrigued by abstract theory and concepts, but I'm most interested in tools and products that work well.

Which brings us to a consideration of OS/2. Over the last few months I've read several books on OS/2 by its authors along with a good share of the thirteen or so volumes plus supplements that come with Microsoft's Software Development Kit. OS/2 is an eclectic operating system, containing features from Unix, the IBM mainframe world, and other environments—one could say too many features. Unix has 64 system calls, of which only about 32 are used extensively. OS/2 has over 300, and with the Presentation Manager, the number of system calls grows towards 700. The sheer quantity of information a competent OS/2 programmer must master seems enormous. Add to this discouragement the matter of the hardware it addresses. If OS/2 (and Windows for that matter) had arrived on schedule two years ago, my reaction would have been very different. It's hard to imagine OS/2 on the 80286 with its 1970s PDP-11 segment hardware limitations competing for very long with multitasking operating systems using the 1980s technology of the 80386 and 680x0 CPUs—paging and 32-bit registers. OS/3, please arrive sooner than later.

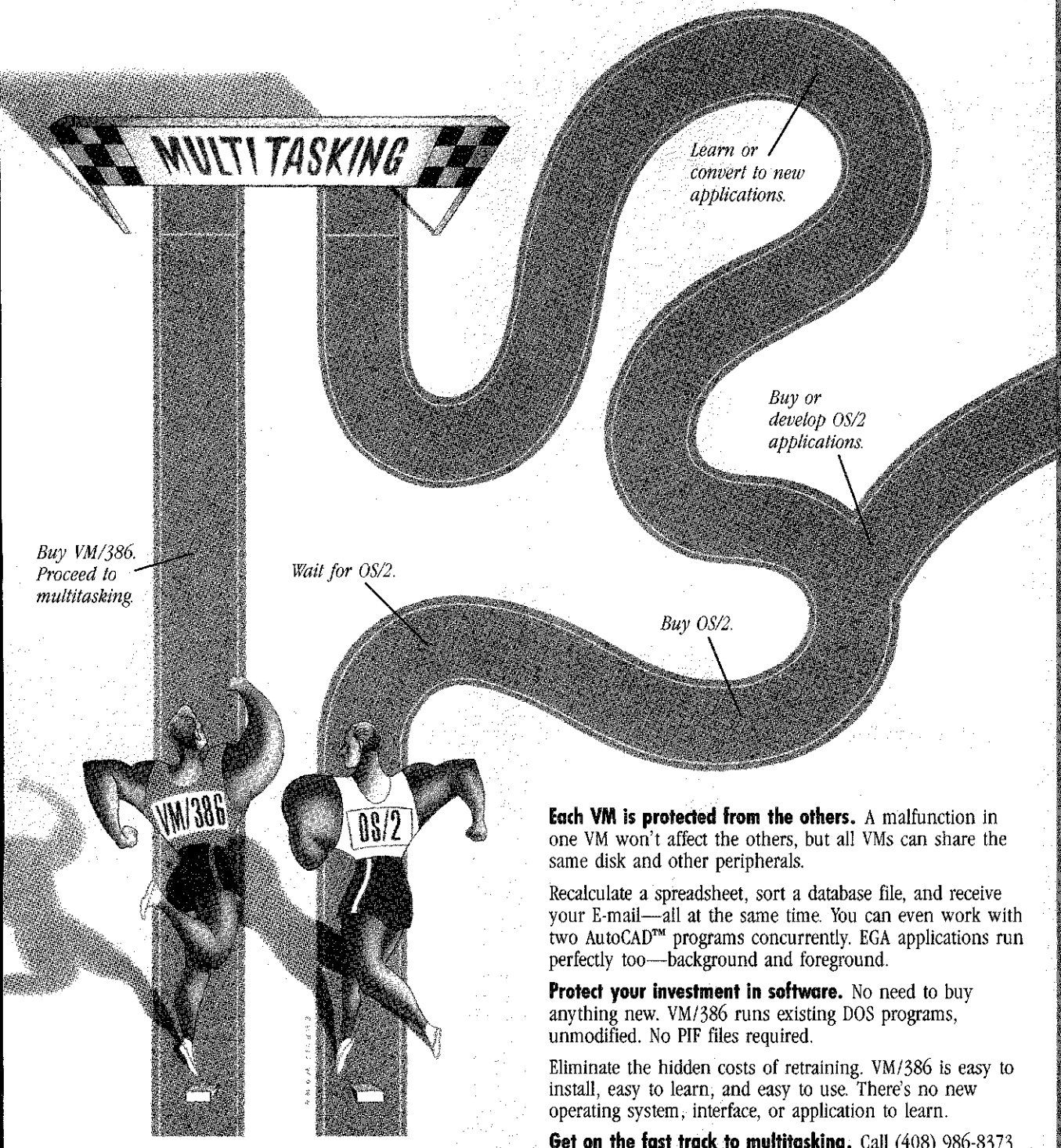
The other hot topic is windowing environments . . . and their politics. X Windows Version 11 Release 2 has been available since the beginning of March. It looks like it will enjoy widespread adoption at least in part as a reaction to politics. Sun's Network File System (NFS) has been so successful that companies on the order of DEC (and soon maybe IBM) find themselves having to license the technology from Sun. DEC, HP, and Tektronix don't look forward to having to license Sun's windowing technology next, so they've supported the X Windows development at MIT. X Windows now consists of about 70 megabytes of source code (about half user contributed). It takes a mere 20 megabytes of disk space just to build the core X Windows distribution.

Then we have Display PostScript from Adobe and the equivalent from Sun. I saw Adobe's version running on a mere 3.5 megabytes of memory.

Finally, we have the IBM Presentation Manager (visually based on Windows 2.0)—already behind schedule, reported to be slow, and now threatened by Apple's lawsuit against Microsoft Windows 2.03. I suppose they could end up going back to the tiled windows of Windows 1.0.

Like OS/2, these windowing environments are large programming projects. You want a stable product when? Do you ever yearn for those lost days of CP/M and 64K?—*Steve* ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



VM/386™. The Fast Track To Multitasking.

Run smart. Run efficient. Run VM/386 on your 386-based PC and start multitasking now! No detours, no waiting, no runaround.

True multitasking. VM/386 uses the virtual 8086 mode of the 80386 processor to create many Virtual Machines (VMs) in one computer.

You can load a different application into each VM. Each VM has up to 640K RAM, plus its own DOS, CONFIG.SYS, AUTOEXEC.BAT and memory-resident programs along with its applications. Tailor each VM to your needs. You have complete control.

Each VM is protected from the others. A malfunction in one VM won't affect the others, but all VMs can share the same disk and other peripherals.

Recalculate a spreadsheet, sort a database file, and receive your E-mail—all at the same time. You can even work with two AutoCAD™ programs concurrently. EGA applications run perfectly too—background and foreground.

Protect your investment in software. No need to buy anything new. VM/386 runs existing DOS programs, unmodified. No PIF files required.

Eliminate the hidden costs of retraining. VM/386 is easy to install, easy to learn, and easy to use. There's no new operating system, interface, or application to learn.

Get on the fast track to multitasking. Call (408) 986-8373 for more information or to order VM/386. Everything else is just the runaround.

System Requirements

80386-based computer such as IBM® PS/2™ Model 80 or COMPAQ® DESKPRO 386® or 80286-based computer with Intel® Inboard™ 386/AT.

One 1.2 Mb (5¼") or one 3½" microfloppy disk drive.

One hard disk drive.

DOS 3.0 or later.

2 Mb RAM recommended.

Supports monochrome, CGA, EGA, VGA, and Hercules™ monitors.

Not copy protected.

IGC
4800 Great America Parkway
Santa Clara, CA 95054
(408) 986-8373

VM/386 is a trademark of IGC. IBM is a registered trademark of International Business Machines Corporation. COMPAQ and DESKPRO 386 are registered trademarks of Compaq Computer Corporation. Intel is a registered trademark and Inboard is a trademark of Intel Corporation. Hercules is a trademark of Hercules Computer Technology. AutoCAD is a trademark of Autodesk, Inc.

Number 169 on the Reader Service Card

PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

That's right. 4 times faster.

Clocking in at over 75,000 lines per minute on a 6MHz IBM AT, OPTASM is four times faster than Microsoft's MASM 5.0. **4 times faster —** that's 400% more throughput!

But speed is only one part of it. OPTASM is nearly 100% compatible with MASM 5.0 (except 386 support).

It is the only single assembler capable of supporting the various incompatibilities between MASM 3, 4 & 5. That makes OPTASM **more MASM compatible than any single version of MASM!**

Other features? OPTASM generates smaller code without ever generating extra NOP's. It automatically handles jumps out of range, up to 15,000 symbols and most of MASM's phase errors. It also boasts a built in MAKE and simplifies segmentation.

That's why we can make our OPTASM challenge: Test OPTASM head to head against MICROSOFT MASM 5.0. Order **both** assemblers with their 30-day guarantees. In a lot less than 30 days, you'll see just how dazzling OPTASM's speed really is. You'll realize that we're compatible, easier to use, and deliver many more important features than MASM. So accept our challenge. Try both assemblers. **Four times faster and more features, too.** We know which one you'll send back.

Write or call us to order or for our detailed brochure.

SLR SYSTEMS

1622 N. Main Street
Butler, PA 16001
412-282-0864
BBS 412-282-2799
Telex 559215
800-833-3061

OPTASM: $195 Guaranteed returnable within 30 days.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Upgrade Your Technology

We're Programmer's Connection, the leading *independent dealer of quality programmer's* development tools for IBM personal computers and compatibles. We can help you upgrade your programming technology with some of the best software tools available.

Comprehensive Buyer's Guide. The CONNECTION, our new Buyers Guide, contains prices and up-to-date descriptions of *over 600 programmer's development tools by over 200 manufacturers*. Each description covers major product features as well as special requirements, version numbers, diskette sizes, and guarantees.

How to Get Your FREE Copy: 1) Use the reader service card provided by this journal; 2) Mail us a card or letter with your name and address; or 3) Call one of our convenient toll free telephone numbers.

If you haven't yet received your copy of the Programmer's Connection Buyer's Guide, act now. Upgrading your programming technology could be one of the wisest and most profitable decisions you'll ever make.

USA 800-336-1166

Canada	800-225-1166
Ohio & Alaska (Collect)	216-494-3781
International	216-494-3781
TELEX	9102406879
FAX	216-494-5260

Business Hours: 8:30 AM to 8:00 PM EST Monday through Friday
Prices, Terms and Conditions are subject to change.
Copyright 1988 Programmer's Connection Incorporated

PROGRAMMER'S Service CONNECTION — QUALITY INTEGRITY SUPPORT PRICE

Established 1984

ORDERING INFORMATION

FREE SHIPPING. Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Call for express shipping rates.

NO CREDIT CARD CHARGE. VISA, MasterCard and Discover Card are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include expiration date and authorized signature.

NO COD OR PO FEE. CODs and Purchase Orders are accepted at no extra cost. No personal checks are accepted on COD orders. POs with net 30-day terms (with initial minimum order of $100) are available to qualified US accounts only.

NO SALES TAX. Orders outside of Ohio are not charged sales tax. Ohio customers please add 5% Ohio tax or provide proof of tax-exemption.

30-DAY GUARANTEE. Most of our products come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

SOUND ADVICE. Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

INTERNATIONAL ORDERS. Shipping charges for International and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. International orders (except Canada), please include an additional $10 for export preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

MAIL ORDERS. Please include your telephone number on all mail orders. Be sure to specify computer, operating system, diskette size, and any applicable compiler or hardware interface(s). Send mail orders to:

Programmer's Connection
Order Processing Department
7249 Whipple Ave NW
North Canton, OH 44720

	List	Ours
386 products		
386 ASM/386 LINK *by Phar Lap*	495	377
386 DEBUGGER *by Phar Lap*	195	129
PC/MOS *All Varieties* ..... *New*	CALL	CALL
NDP C-386 *by MicroWay* ..... *New*	595	529
NDP ForTran-386 *by MicroWay* ..... *New*	595	529
blaise products		
ASYNCH MANAGER *Specify C or Pascal*	175	135
C TOOLS PLUS/5.0	129	99
LIGHT TOOLS *for Datalight C* ..... *Sale*	100	55
PASCAL TOOLS/TOOLS 2	175	135
RUNOFF *Text Formatter*	50	45
Turbo ASYNCH PLUS/4.0	129	99
Turbo C TOOLS	129	99
Turbo POWER TOOLS PLUS/4.0	129	99
VIEW MANAGER *Specify C or Pascal*	275	199
borland products		
Paradox 1.1 *by Ansa/Borland*	495	359
Paradox 2.0 *by Ansa/Borland*	725	525
Paradox 386 *by Ansa/Borland* ..... *New*	895	CALL
Paradox Network Pack *by Ansa/Borland*	995	725
Quattro: The Professional Spreadsheet	247	179
Sidekick Plus ..... *New*	200	139
Turbo Basic Compiler	100	68
Turbo Basic Database Toolbox	100	68
Turbo Basic Editor Toolbox	100	68
Turbo Basic Telecom Toolbox	100	68
Turbo C Compiler	100	68
Turbo Pascal	100	68
Turbo Pascal Database Toolbox	100	68
Turbo Pascal Developer's Toolkit	395	285
Turbo Pascal Editor Toolbox	100	68
Turbo Pascal Gameworks Toolbox	100	68
Turbo Pascal Graphix Toolbox	100	68
Turbo Pascal Numerical Methods Toolbox	100	68
Turbo Pascal Tutor	70	49
Turbo Prolog Compiler	100	68
Turbo Prolog Toolbox	100	68
c language		
C Programmer's Toolbox Combo *by MMC AD* ..... *New*	130	115
C Programmer's Toolbox *Specify Vol I or II* ..... *New*	80	69
C-terp *by Gimpel, Specify compiler*	298	219
for XENIX	498	379
dBx *dBASE to C Translator by Desktop AI*	550	429
with Source Code	950	729
DC88 *by C Ware* ..... *New*	99	89
DC88 *with Large Case Option by C Ware* ..... *New*	178	159
DMS Resident-C *by American Software Intl*	150	139
Eco-C88 Modeling Compiler *by Ecosoft*	100	69
Lattice C Compiler *from Lattice* ..... *New Version*	450	239
PC Lint *by Gimpel Software*	139	99
Turbo C-terp *for Turbo C by Gimpel*	139	119
Vitamin C *by Creative Programming*	225	149
VC Screen *Forms Designer*	*100	79
vLIB *by Pathfinder Associates* ..... *New*	99	89
with Source Code ..... *New*	149	129
WATCOM C6.0 *by Waterloo Computer* ..... *New*	295	249
WKS LIBRARY *by Tenon Software*	89	79
laboratory microsystems products		
CFORTH	200	149
FORTH/83 Metacompiler *Specify Target*	750	599
PC/FORTH	150	109
PC/FORTH+	250	188
Programmer's Package #1	250	199
Programmer's Package #2	350	279
Programmer's Package #3	500	399
UR/FORTH *Also Available for OS/2*	350	258
UR/FORTH Libraries	500	395

Peabody Pop-Up Reference Utility by Copia International

Peabody is a fast and flexible on-line reference utility. It provides instant, accurate and complete language information in pop-up frames at the touch of a key. With Peabody, you can select general topics from a structured subject menu, or use Peabody's hyperkey to get instant help for the keyword closest to the cursor.

Peabody *Specify 1 Database*	100	89
Additional Databases:		
Microsoft C v 5.1 Database	50	45
Microsoft Macro Assembler v 5.1 Database	50	45
MS DOS v 3.3 Database	50	45
Turbo C v 1.5 Database	50	45
Turbo Pascal v 3.0 Database	50	45
Turbo Pascal v 4.0 Database	50	45
Peabody Engine Only	50	45
lahey computer products		
F77L-EM/16 ..... *New*	695	CALL
F77L-FORTRAN Compiler ..... *New*	477	CALL
Lahey Personal Fortran 77 ..... *New*	95	CALL
metagraphics products		
MetaWINDOW *No Royalties*	195	159
MetaWINDOW/PLUS	275	229
QuickWINDOW/C *for Microsoft QuickC*	95	79
TurboWINDOW/C *for Borland Turbo C*	95	79
TurboWINDOW/Pascal *for Borland Turbo Pascal*	95	79
metaware products		
High C ..... *New*	595	529
386 Version ..... *New*	895	789
Professional Pascal ..... *New*	595	529
386 Version ..... *New*	895	789
microsoft products		
Microsoft C Compiler *w/CodeView* ..... *New Version*	450	285
Microsoft COBOL Compiler *with COBOL Tools*	700	439
Microsoft FORTRAN *Optimizing Comp* ..... *New Version*	450	285
Microsoft Macro Assembler ..... *New Version*	150	99
Microsoft Mouse *All varieties*	CALL	CALL
Microsoft OS/2 Programmer's Toolkit ..... *New*	350	CALL
Microsoft Pascal Compiler ..... *New Version*	300	189
Microsoft QuickBASIC	99	69
Microsoft QuickC	99	69
Microsoft Windows	99	69
Microsoft Windows 386	195	129
Microsoft Windows Development Kit	500	319
periscope products		
Periscope I *with Board*	345	275
Periscope II *with NMI Breakout Switch*	175	139
Periscope II-X *Software only*	145	105
Periscope III *8 MHz version*	1095	875
Periscope III *10 MHz version*	1195	949
other products		
ACTOR *by Whitewater Group*	495	419
Brief *by Solution Systems*	195	CALL
Dan Bricklin's Demo II *by Software Garden*	195	155
Desqview *from Quarterdeck*	130	115
DMS Resident-ASM *by American Software Intl*	150	139
DMS Screen Master *American Software Intl* ..... *New*	200	185
Finally *by Komputerwerk*	99	85
GRASP *by Paul Mace Software* ..... *New*	99	85
Heap Expander *by The Tool Makers* ..... *New*	60	55
HI-Screen XL *by Softway* ..... *New*	149	119
Interactive EasyFlow *by HavenTree Software*	150	125
Mace Utilities *Paul Mace Software*	99	85
MASTER*KEY *by Sharpe Systems* ..... *New*	80	69
OPT-Tech Sort *by Opt-Tech Data Proc*	149	99
OPTASM *by SLR Systems*	195	169
Pascal-2 *by Oregon Software*	395	289
QBase *Relational DB by Crescent Software*	99	89
QuickPak *by Crescent Software*	69	59
Quilt Computing Combo *QMake & SRMS*	250	199
risC *by IMSI*	80	65
SourceTools *by Oregon Software* ..... *New*	595	429
Taskview *by Sunny Hill Software*	80	55
Tmark *by Tangent Designs*	80	69
Turbo Programmer *by ASCII* ..... *New*	289	229
TurboHALO *from IMSI*	95	75
Vedit Plus *by CompuView*	185	128
XenoCopy-PC *by XenoSoft*	80	69

CALL for Products Not Listed Here

Number 146 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Q. How many programmers does it take to maintain a MAKE dependency file?

A. NONE! *If you use* VersiMAKE™

VersiMAKE™ is a full-featured MAKE utility that includes:

- **Dependency Generation**
  Derives your system's dependencies, through analysis of its C and MASM source files. Say goodbye to manual maintenance of MAKE dependency files!
- **Wild Card File Name Matching**
  Analyzes an entire collection of source files with a single statement.
- **Nested Include File**
  Handles standard C and MASM "include" conventions, and the INCLUDE environment variable.
- **Powerful Macro Facilities**
  Built-in macros, user-defined macros, and environment variables.
- **Analytical Reports**
  Shows the entire Include file hierarchy for each source file analyzed, and all of the parent source files for each Include file.

Q. How many programmers does it take to trace a symbol thru your system?

A. ONE! *If you use* VersiCREF™

VersiCREF™ is a unique utility that creates a Master Cross-Reference of your entire system.

- **Multi-Lingual**
  Handles C, assembler, or both.
- **Flexible**
  File names with line numbers, or file names alone. Global and local symbols, or globals alone.
- **Powerful**
  Easily handles systems containing 100 source files or more.

VersiMAKE™	$125
VersiCREF™	$75
Both	$150

(Outside U.S., add $5 for shipping and handling)

800-334-4096
(In NJ, 609-871-0202)
MC/VISA/AMEX accepted

SUMMIT INFORMATION SYSTEMS, INC.
73 East Lane, Willingboro, NJ 08046

Number 158 on the Reader Service Card

"Developing my application in C would have taken 6 months to a year, but in Actor it took 2 months." —Brian Fenske, Boeing Commercial Airplane Company

"To C or not to C..." ACTOR

Actually, you don't have to make the choice. Once C was ideal for all PC programming. But it has been complicated by windowing and graphical interfaces. Now windows development with C is difficult, time-consuming and error-prone. You need a new language that simplifies windows programming. Introducing Actor®.

Actor is the first interactive object-oriented language made for commercial development. Its powerful browsers, inspectors and debuggers give you more insight into a windowing environment than C ever will. But your C work is not lost. C libraries can be linked to Actor. Plus, its procedural syntax is easy for C programmers to learn.

Actor comes with windowing classes built in. Customize Actor's classes to create stand-alone windowing applications. And objects give you another layer of independence for a smooth transition to OS/2 and Presentation Manager. It's the quickest and easiest way to write a windowing program.

"You can write Windows programs much faster with Actor than with C or assembly language."
—PC Magazine, June 9, 1987

Tech Specs

New Release Ver. 3.0 or 1.1

- Runs with Microsoft Windows 1.04, 2.0 and 386. Extended memory under 2.0 and 386.
- Pure, single-inheritance object-oriented language, incrementally compiled.
- Dynamic linking to C, Pascal, Assembler, or Fortran libraries. Pass data in C structures.
- Pascal and C-like syntax.
- Programming tools: Browser, Inspector, Debugger, File Editor.
- Full access to MS-Windows systems calls, multitasking, and DDE.
- Fast device-independent graphics: lines, shapes, icons, cursors, bitmaps, metafiles, Turtle graphics, sample control language using YACC.
- 150 classes, 1500 functions, fully extensible.
- Window styles: tiled, overlapping, popup, child, edit, dialogs. Controls: list boxes, scroll bars, buttons, check boxes.
- Data structures: stacks, arrays, queues, lists, dictionaries, sets, sorting, hashing, intervals.
- AI support: frames, symbols, dictionaries, lists, symbolic programming, functional arguments. Parsing and lexical analysis YACC compatible.
- String manipulation: substring, concat, append, insert, remove, search.
- 643-page manual includes tutorial and reference.
- No license fees. Generates stand-alone applications.
- Fastest interactive OOL available.
- Fast incremental garbage collector.

Actor $495 • Academic price $99 • Academic site license $99 • Manuals for site license $35 • New! Language Extension $99 • Shipping $5 US, $25 Int'l

The Whitewater Group
Technology Innovation Center
906 University Place, Evanston, Illinois 60201
(312) 491-2370

Actor is a registered trademark of The Whitewater Group, Inc.

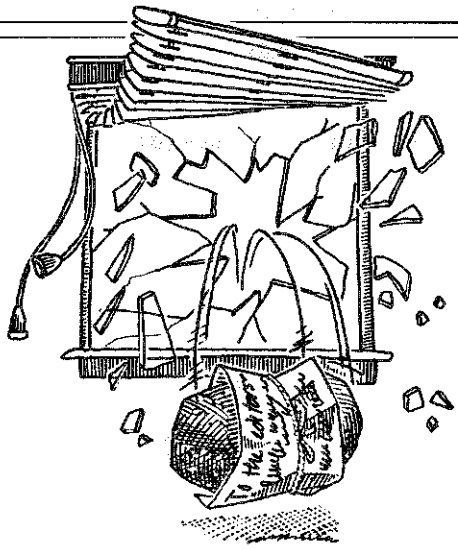
Number 73 on the Reader Service Card

MOVING? BE SURE TO KEEP THOSE PJs FLOWING . . .

My old ZIP code is: _____
My New Address is:
Name _____
Company _____
Name _____
City _____ State__ ZIP_____

MAIL TO: *Programmer's Journal*, P. O. Box 30160, Eugene, OR 97403

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Recover Your Losses

Your January/February 1988, issue contains a letter from Mr. Grey Staples describing the terrible result of using an IBM PC XT diagnostic disk to park the heads of a PC AT hard disk. Having wrestled with the problem several times, I would like to report that it is partially, if not wholly, recoverable.

Mr. Staples' description of the problem is correct. The very first track of the AT's disk is smashed in such a way that it is no longer recognizable by the BIOS POST routine. The BIOS therefore considers the drive to be inaccessible and unusable. Track 0 must be reformatted, and the boot sector and other low disk tables restored. An attempt can then be made to salvage the various user files.

I use the Mace Utilities and a package titled HTEST/FORMAT (the latter from Kolod Research, Inc.—both available from Paul Mace Software, 800/523-0258). Recovery is as outlined by Mr. Staples: Reformat track 0 (only) using HFORMAT, run FDISK, and FORMAT to rebuild the low disk tables, then recover the user data as though from an inadvertent FORMAT operation. The Mace Utilities was the first commercial product to offer an Unformat function. Depending on various conditions, it can be quite successful. If the Mace RXBAK program was installed on the system prior to the accident, recovery should be nearly complete. If it was not and many of the files were fragmented, recovery may range from tedious to impossible.

I have used these two packages successfully (and, sometimes, unsuccessfully) in this and other hard disk misadventures. I consider them to be indispensable components of my PC "Hero" kit (the bundle of software I carry on trouble calls). What's more, the two products contain a wealth of other disk maintenance and support programs, not the least of which is a head parking utility that can be used safely on any system.

For that matter, does anyone know of a public domain, that is, zero cost, head park utility? 'Twould seem that such a program could be researched and written in a day or two.

Konrad M. Wiley
Technology Analyst
Nashville, TN

We'll put just such a park utility on the PJ *BBS's and the Issue Disk for Volume 6.3.—Ed.*

You're Right, But . . .

Regarding the ability of a Paul Mace program to perform a low-level format of just track 0, I stand corrected. It should not be necessary to Unformat the DOS partition after a successful FDISK rebuild of the Master Boot sector. The partition should be intact, with only track 0 suffering the damage.

But let's consider some complications. In this repair scenario, FDISK starts from scratch and decides whether to use 12- or 16-bit FAT entries, a decision involving the partition size and version of DOS. I've dealt with an upgraded PC with a 20 Meg drive installed, originally formatted using DOS 2.1 and having 12-bit FAT entries. Then the drive was reformatted with DOS 3.1 and the partition rebuilt, but FDISK (3.1) was not performed. We still had 12-bit FAT entries, and the Partition Boot sector was in the next sector after the Master Boot, rather than on the next track or cylinder. Therefore, trashing track 0 has taken out the beginning 16 sectors of the partition as well. Our DOS 3.1 version of Format has also given us an 8-sector FAT. All we have left after losing track 0 is the last (eighth) sector of the second FAT copy. Now we pull out the Mace Unformat and try it.

Grey Staples
Information Repair Services, Inc.
Scottsdale, AZ

And Then There's . . .

As a sysop and frequenter of several BBS's, I'm concerned about the recurrent phenomenon of the Trojan horse. The newest is VIRUS, a piece of code that alters a disk's COMMAND.COM and/or hidden DOS files. VIRUS manages this patching of itself to the system files in such a way that the size of the file and other noticeable traits such as time and date are not affected (some accounts have claimed that the date of the affected file changes).

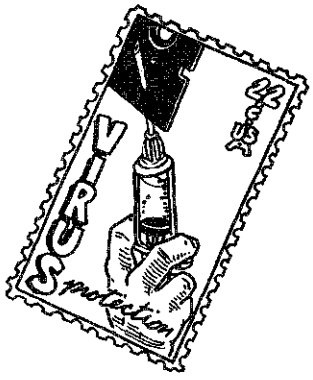
If all we had here was a patch that did nothing more than relocate itself, we could all marvel at the author's legerdemain, but this virus wipes out all the files on all disks on the computer system, not just the FAT. It sets a counter or a timer, and when the magic value is reached, poof! All gone! All disks that came in contact with the infected disk before it died from the virus infection are also infected—and infecting!

FLU_SHOT.ARC and FLUSHOT2.ARC were supposed to detect a potential alteration of the DOS files and warn you. But some of the creeps who write Trojan horses have successfully targeted FLU_SHOT for alteration. FLU_SHOT3.ARC is so far a valid program. Its author runs a BBS (212/889-6438). Be sure to ask the sysop of your local BBS if he or she obtained the file from the author.

This virus could sweep from BBS to BBS via patches onto shareware programs, meaning that shareware authors and sysops would have to form mutual networks to ensure safe computing. And this would destroy the vitality of the current state of affairs. No sysop has the time to get files directly from the source. The network survives today because of callers who upload new files.

As programmers, perhaps you can help me understand why someone would write code such as VIRUS. Drop me a brief note through *PJ* or on the BBS and share your views. How can honest authors of shareware protect themselves and BBS's from such low-lifes?

Gerry Power
Turbo Source Search
Scituate, MA



This One's Easy (for Michael Abrash)

Has *PJ* published any articles and/or code showing how to tell what mode an EGA card is in? I'm trying to figure out whether the EGA is in 25- or 43-line mode without resetting/re-initializing the EGA card so that I can restore the original mode when exiting a program.

Pete Smithem
Spring Valley, NY

Michael Replies

All you need to do to figure out whether you're in 25- or 43-line mode is check the

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C CODE FOR THE PC

source code, of course

Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	$400
CQL Query System (SQL retrievals plus windows)	$325
GraphiC 4.1 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
Barcode Generator (specify Code 39 (alphanumeric), Interleaved 2 of 5 (numeric), or UPC)	$300
Greenleaf Data Windows (windows, menus, data entry, interactive form design)	$295
Aspen Software PC Curses (System V compatible, extensive documentation)	$250
Vitamin C (MacWindows)	$200
TurboTEX (TRIP certified; HP, PS, dot drivers; CM fonts; LaTEX)	$170
Essential resident C (TSRify C programs, DOS shared libraries)	$165
Essential C Utility Library (400 useful C functions)	$160
Essential Communications Library (C functions for RS-232-based communication systems)	$160
Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	$150
Greenleaf Functions (296 useful C functions, all DOS services)	$150
OS/88 (U**x-like operating system, many tools, cross-development from MS-DOS)	$150
Turbo G Graphics Library (all popular adapters, hidden line removal)	$135
American Software Resident-C (TSRify C programs)	$130
PC Curses Package (full System V, menu and data entry examples)	$120
CBTree (B+tree ISAM driver, multiple variable-length keys)	$115
Minix Operating System (U**x-like operating system, includes manual)	$105
PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
The Profiler (program execution profile tool)	$100
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	$100
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	$100
Wendin Operating System Construction Kit or PCNX, PCVMS O/S Shells	$95
C Windows Toolkit (pop-up, pull-down, spreadsheet, CGA/EGA/Hercules)	$80
Professional C Windows (windows and keyboard functions)	$80
JATE Async Terminal Emulator (includes file transfer and menu subsystem)	$80
MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
ME (programmer's editor with C-like macro language by Magma Software)	$75
Professional C Windows (lean & mean window and keyboard handler)	$70
WKS Library (C program interface to Lotus 1-2-3 program & files)	$65
Quincy (interactive C interpreter)	$60
EZ_ASM (assembly language macros bridging C and MASM)	$60
PTree (parse tree management)	$60
HELP! (pop-up help system builder)	$50
Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
Heap Expander (dynamic memory manager for expanded memory)	$50
Make (macros, all languages, built-in rules)	$50
Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	$50
Coder's Prolog (inference engine for use with C programs)	$45
C-Help (pop-up help for C programmers ... add your own notes)	$40
Biggerstaff's System Tools (multi-tasking window manager kit)	$40
CLIPS (rule-based expert system generator, Version 4.0)	$35
TELE Kernel or TELE Windows (Ken Berry's multi-tasking kernel & window package)	$30
Clisp (Lisp interpreter with extensive internals documentation)	$30
Translate Rules to C (YACC-like function generator for rule-based systems)	$30
6-Pack of Editors (six public domain editors for use, study & hacking)	$30
ICON (string and list processing language, Version 6 and update)	$25
LEX (lexical analyzer generator)	$25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
AutoTrace (program tracer and memory trasher catcher)	$25
C Compiler Torture Test (checks a C compiler against K & R)	$20
Benchmark Package (C compiler, PC hardware, and Unix system)	$20
TN3270 (remote login to IBM VM/CMS as a 3270 terminal on a 3274 controller)	$20
A68 (68000 cross-assembler)	$20
List-Pac (C functions for lists, stacks, and queues)	$20
XLT Macro Processor (general purpose text translator)	$20
Data	
WordCruncher (text retrieval & document analysis program)	$275
DNA Sequences (GenBank 52.0 including fast similarity search program)	$150
Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	$60
Webster's Second Dictionary (234,932 words)	$60
U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
USNO Floppy Almanac (high-precision moon, sun, planet & star positions)	$20
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409 USA

acwlinfo@uunet.uu.net

Voice: (512) 258-0785
BBS: (512) 258-8881
FidoNet: 1:382/12

Free shipping on prepaid orders **For delivery in Texas add 7%** **MasterCard/VISA**

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

setting of the BIOS Rows variable at 0:484 (or 40:84 if you prefer to think of it that way). If Rows is 18h (24 decimal), you're in 25-line mode; if Rows is 2Ah (42 decimal), you're in 43-line mode.

Michael Abrash
Mountain View, CA

Letdown

Frank Greco's article "Time Is on My Side . . ." in Volume 6.1 (January/February 1988) was a letdown. He's usually much better. He completely ignored the high speed clock interrupt approach, or reprogramming the 8254 and handling the 18.2 milliseconds tick in the timer procedure.

Anonymous
Van Nuys, CA

Ready to Download

I enjoyed Frank Greco's article "Time Is on My Side . . . Yes, It Is" in *PJ*, 6.1. Frank said that if anyone would like to see the System V Unix version of the macros to let him know. Well, I'm letting him know. Also, would you assume that the STOPWATCH program advertised on page 61 of the same *PJ* issue works in the manner Frank outlined?

Phil Pistone
Park Ridge, IL

I Can't Get No Satisfaction

Anonymous, what is the "high speed clock interrupt approach"?

Phil, since several other people wanted a higher resolution version of my timing function/macro, I added some reprogramming to the timer chip and a BIOS clock-tick fix-up kludge (take that!) to time some short-duration events. I also altered my header file (timer.h) to portably handle System V Unix, Berkeley Unix, *and* PC/MS-DOS! See the Redirections column this month for a listing of that header file and a sample C program. Thanks for the feedback, everybody.

Frank Greco
Fanwood, NJ

The Answer is Yes

STOPWATCH can indeed be used to measure code performance in a way similar to that described in Frank Greco's article, but STOPWATCH returns its readings in *microseconds* rather than in eighteenth of a second ticks and enables programmers to make timing measurements without modifying their code.

To time the **kbhit()** C library routine using STOPWATCH and a Microsoft debugger, for example, you could run up until the line containing the call, then request that the debugger go and break at the next line. If the go to breakpoint request is terminated by a Ctrl-M rather than Enter, STOPWATCH presents the following results when run on a 6MHz AT: PROG 13, BIOS 140, DOS 543, DISK 0. This means that the high level code took 13 microseconds but invoked DOS and BIOS code that required a combined 683 microseconds—surprisingly high for a routine that merely checks for an unprocessed keystroke. STOPWATCH operates passively on a single execution of the code under test, so debugging may continue with or without further STOPWATCH measurement.

Michael Liebert
Custom Real-Time Software, Inc.
West Caldwell, NJ

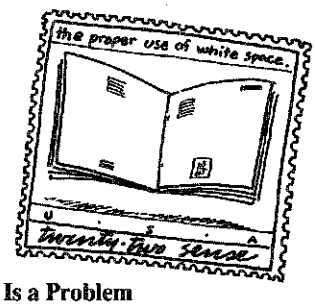
Minor Errors

Re "Inside Turbo Pascal 4.0" in the March/April 1988, issue: Two minor errors in the otherwise fine article could cause some problems for readers. On page 39, John Newlin states, "The pointer type can point to anything and is initialized to nil (0000:0000)." I'm afraid this is wishful thinking—uninitialized pointers are like any other uninitialized variables, as this tiny program demonstrates:

```
PROGRAM foo;
VAR P : pointer
BEGIN
  WriteLn(LongInt(P));
  P := NIL
  WriteLn(LongInt(P));
END
```

On page 40, John states that pressing the Alt-F1 key with the cursor on an identifier will bring up help for that identifier. In fact, Ctrl-F1 is the key that has this effect. Alt-F1 brings up the most recently viewed help screen. The effect is similar enough that a user might imagine the help key to be broken.

Neil J. Rubenking
San Francisco, CA



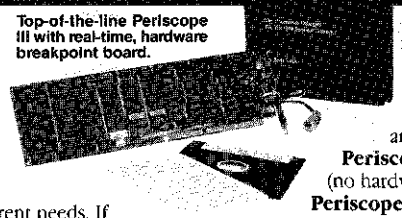
This Is a Problem

I have what I hope is a rare problem with your publication, with *PJ*, Volume 5.6, November/December 1987. Pages 12 and 13, which should contain more of Christian Walker's piece, and pages 68 and 69, which should contain more of M. Steven Baker's work, are mute. They speak only in loud silences . . . fields of uninterrupted white on white.

White is beautiful—in the undisturbed stillness of a blank canvas, not even a texture interferes with the inspired images one's mind can place there to dance, lie in reflective repose, or act out some dramatic crea-

PERISCOPE™ POWER

...Keeps you going full steam ahead when other debuggers let you down. With four models to pick from, you'll find a Periscope that has just the power you need.



Top-of-the-line Periscope III with real-time, hardware breakpoint board.

. . . Start with the model that fits your current needs. If you need more horsepower, upgrade for the difference in price plus $10! And don't worry about having a lot more to learn . . . Even when you move to the most powerful model, Periscope III, an extra dozen commands are all that's involved.

Periscope's software is solid, comprehensive, and flexible. It helps you debug just about any kind of program you can write . . . thoroughly and efficiently. Periscope's hardware adds the power to solve the really tough debugging problems.

Periscope requires an IBM PC, XT, AT, or close compatible (Periscope III requires hardware as well as software compatibility); DOS 2.0 or later; 64K available memory; one disk drive; an 80-column monitor.

Periscope I includes a half-length board with 56K of write-protected RAM; break-out switch; software and manual for $345.

Periscope II includes break-out switch; software and manual for $175.

Periscope II-X includes software and manual (no hardware) for $145.

Periscope III includes a full-length board with 64K of write-protected RAM, hardware breakpoints and real-time trace buffer; break-out switch; software and manual. Periscope III for machines running up to 8 MHz is $995; for machines running up to 10 MHz, $1095.

Call Toll-Free for free information or to order your Periscope today!

MAJOR CREDIT CARDS ACCEPTED. **800-722-7006**

The PERISCOPE Company, Inc.
1197 PEACHTREE ST.
PLAZA LEVEL
ATLANTA, GA 30361
404/875-8080

Number 6 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

tion of the subconscious projected thereon. What a wonder this unsullied expanse of blankness. Stillness of the mind and body puts the soul in touch with the rest of creation, achieving the oneness known so intimately, yet continually forgotten. One, seeking, rarely finds what this unguarded moment inspired. Reaching out to grasp the thing causes it to flee to just beyond the farthest reach. Yet, here, in a magazine, I've found not one, not two, but four grand opportunities to make contact with . . .

Wait, one moment . . . I don't turn to magazines as a means of practicing ZEN. I seek the awareness inspired by shapely black glyphs on a field of omnireflective white, pleasingly arranged, well ordered, highly symbolic and significant of practical matters, conveying information about computers and related topics. In magazines, I do not often seek the depth of the symbolic virginity of the medium on which these glyphs are placed, do not often see the lines and squiggles as more than representations of practical communications, do not often speculate on the higher-order messages one less concerned with the immediacy of this world and tangible objects might perceive.

That which I seek is to obtain copies of the text omitted from my copy of your magazine. If you would be so kind as to supply information as to how I might obtain copies of the missing pages, I would be grateful.

David Marciel
Sacramento, CA

David, by all means and the sooner the better! Send in your old, mantric issue of PJ, and we'll replace it with the mundane version. Other readers similarly stricken, please do the same.—Ed.

A Heap of Trouble?

Has anybody else out there using Borland's new gleam machine TP4 seen a *degradation* in performance of their favorite Pascal masterpiece? If so, you're not alone. After upgrading a large (24,000 lines) program from TP 3.02 to TP 4.0 (a task greatly facilitated by the new separate compilation capabilities), I found that instead of the improvement in speed I had seen with other upgraded programs, the final product was three to five times slower. Problems that had previously taken ten minutes to run with TP3 were now taking forty-five minutes.

Borland Technical Support provided no clues. At the suggestion of our master programmer, I investigated the performance of Turbo 4's new heap manager. Our application does lots of **news** and **disposes** on small objects, and sure enough, this is enough to send TP4's new heap management into a frenzy. With the 1 byte granularity, the free list gets very large very quickly and probably becomes an albatross to update. If you've noticed a dramatic decrease in the speed of programs compiled with TP4, this could be it.

I have included a small program that badly fragments the heap by allocating and disposing of small objects in a random fashion. Note the time comparisons between TP3 and TP4.

I sent this problem to Borland Tech Support, whose reply was to suggest obvious techniques that would involve serious rewrite on our part. I just want the old heap manager back! I love the new product and all its capabilities—I just wish it would produce something we could send people. Tech Support said they would send the problem on to the TP4 developers, so I haven't given up hope.

Name withheld upon request
Princeton, NJ

```
program fragheap;

{ this program demonstrates the degraded performance in TURBO 4.0's     }
{           heap manager over that of TURBO 3.02A                       }
{ things can be made very bad very quickly by increasing array_length   }
{                                                                       }
{ allocation and disposal of medium to small objects in a random        }
{ fashion probably results in lots of overhead to keep the new          }
{ Freelist updated.                                                     }
{                                                                       }
{ Times on 8088 Leading Edge Model D :      }
{    TURBO 3 - 7.34 s                       }
{    TURBO 4 - 4 min 25.59 s  ****  }
{                                           }
{ Times on IBM PC/AT :                      }
{    TURBO 3 - 2.64 s                       }
{    TURBO 4 - 1 min 55.72 s  ****  }


const
   array_length = 3000;
   five_char_string = '12345';
   nine_char_string = '123456789';

type
   grabbptr = ^grabarray;
   grabarray = array[1..200] of array[1..200] of byte;

   fbo_ptr = ^five_byte_obj;
   nbo_ptr = ^nine_byte_obj;

   five_byte_obj = array[1..5] of char;
   nine_byte_obj = array[1..9] of char;

var
   struct1 : array[1..array_length] of fbo_ptr;
   struct2 : array[1..array_length] of fbo_ptr;
   struct3 : array[1..array_length] of nbo_ptr;

   dumarray1,
   dumarray2 : grabbptr;

   i : integer;

begin  { fragheap }

{ allocate more than 64K of memory right off }
{ we're working on top of an already large dynamically allocated data set }
{ the time results are the same whether this is done or not }
new(dumarray1);
new(dumarray2);

{ now try to fragment heap }
{ but not too hard }

{ alternately allocate 5 byte objects and 9 byte objects }
for i := 1 to array_length
do begin
   new(struct1[i]);  new(struct3[i]);
   struct1[i]^ := five_char_string;
   struct3[i]^ := nine_char_string
   end;

{ alternately dispose 9 byte obj - alloc 5 byte obj - dispose 5 byte obj }
{ things are somewhat worse if we go bottom up but top towards bottom       }
{ more closely approximates normal situations and what our program requires }
for i := array_length downto 1
   do begin
   dispose(struct3[i]);
   new(struct2[i]);
   struct2[i]^ := five_char_string;
   dispose(struct1[i])
   end

end.   { fragheap }
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Dear Anonymous
Turbo Pascal Programmer

The example program you provided, FragHeap, is not a good test of both Turbo Pascal 3's (TP3) and Turbo Pascal 4's (TP4) heap managers because it fragments the TP4 heap but does not fragment TP3's. As you discovered, the two compilers use different techniques for storing the free list and your FragHeap example just happens to work nicely on one and not on the other. Run Heap3 under TP3 and run Heap4 under TP4.

The first example (Heap3), like your FragHeap program, takes advantage of TP3 heap management side effects. The second example (Heap4) similarly benefits from TP4 heap architecture. Here are the timings for the two programs (in seconds) when run on an 8 MHz IBM AT:

	TP3	TP4
Heap3	0.6	9.8
Heap4	10.7	0.4

We used a different heap manager in TP4 because we received many requests for a more memory-conservative heap from Turbo Pascal users. For most applications, the TP4 heap performance is as fast as TP3's and it is much more space efficient. TP3 always allocates blocks as a multiple of eight bytes whereas TP4 allocates only exactly as much as is required. Referring to the Heap3 program above—TP3 will allocate 16 bytes for each P9 pointer, and 8 bytes for each P5 pointer. Compared to TP4, this requires almost twice as much heap space.

Though the Heap4 example shows that it's easy enough to provide a counter-example, it doesn't help solve *your* problem. To do that would obviously require a careful look at how your application affects the heap. However, two suggestions can be offered:

1. In program Heap3 above, split the "fragmentation loop" into two for loops—one that first allocates all Q5 pointers, followed by a second loop that disposes all P5 and P9 pointers. In this case, both TP3 and TP4 run in the half-second range (TP4 is a little faster). Of course, the peak use of heap space is a little higher, but due to its more efficient allocation method, the TP4 version of the rewritten program will still use less heap space than the TP3 original. Perhaps a similar change to your big application is possible and will solve your problem?

2. If not, try using HEAP3.TPU. After we saw your letter, we wrote and tested a TP4 unit that does TP3-style heap management. We are sending it to you via the editor (it's available to the public in DL 4 on the BorProgB SIG on CompuServe). You'll need to convert your New(p) and Dispose(p) calls to GetMem(p, SizeOf(p^)) and FreeMem(p, SizeOf(p^)), but no other modifications should be necessary.

We hope this addresses your concerns and solves your problem. If not, please feel free to contact our Tech Support department again. We've given them more training in this area, and they're ready and able to help.

Borland International
Scotts Valley, CA

```
program Heap3; {$R-}
{ Compile and run under both Turbo Pascal 3.0 and 4.0.
  Runs faster under 3.0.
}

const
  ArraySize = 1000;

type
  Byte5 = array[1..5] of Byte;
  Byte9 = array[1..9] of Byte;

  Ptr5 = ^Byte5;
  Ptr9 = ^Byte9;

var
  P5: array[1..ArraySize] of Ptr5;
  P9: array[1..ArraySize] of Ptr9;

  Q5: array[1..ArraySize] of Ptr5;

  I: Integer;

  Timer: Integer absolute $40:$6C;
  Start: Integer;

begin
  WriteLn('Start');
  Start := Timer;

  for I := 1 to ArraySize do          { allocation loop }
  begin
    New(P5[I]);
    New(P9[I]);
  end;

  for I := ArraySize downto 1 do      { fragmentation loop }
  begin
    Dispose(P9[I]);
    New(Q5[I]);
    Dispose(P5[I]);
  end;

  Writeln('Done in ',(Timer-Start)/18.2:0:1,' seconds.');
end.
```

```
program Heap4; {$R-}
{ Compile and run under both Turbo Pascal 3.0 and 4.0.
  Runs faster under 4.0.
}

const
  ArraySize = 1000;

type
  Byte3 = array[1..3] of Byte;
  Byte5 = array[1..5] of Byte;
  Byte7 = array[1..7] of Byte;
  Byte9 = array[1..9] of Byte;

  Ptr3 = ^Byte3;
  Ptr5 = ^Byte5;
  Ptr7 = ^Byte7;
  Ptr9 = ^Byte9;

var
  P5: array[1..ArraySize] of Ptr5;
  P7: array[1..ArraySize] of Ptr7;

  Q3: array[1..ArraySize] of Ptr3;
  Q9: array[1..ArraySize] of Ptr9;

  I: Integer;

  Timer: Integer absolute $40:$6C;
  Start: Integer;

begin
  WriteLn('Start');
  Start := Timer;

  for I := 1 to ArraySize do          { allocation loop }
  begin
    New(P5[I]);
    New(P7[I]);
  end;

  for I := ArraySize downto 1 do      { fragmentation loop }
  begin
    Dispose(P5[I]);
    New(Q3[I]);
    New(Q9[I]);
    Dispose(P7[I]);
  end;

   Writeln('Done in ',(Timer-Start)/18.2:0:1,' seconds.');
end.
```

Continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Critique of Volume 6.2

Several comments on letters and articles in the March/April 1988, *Programmer's Journal*.

First, I thoroughly enjoyed John Newlin's review of Turbo Pascal Version 4. It was fun to compare it with my article on the same subject to appear in the April *PC Tech Journal*. As one might expect, he covered some things that I didn't and vice versa. As one might infer from the content of the following paragraph, Mr. Newlin is kinder to TP4 than I.

Next, I am quite sympathetic with Jon Greenblatt's plight in having to squeeze out the last bit of speed from programs written in Turbo Pascal. I have to care for about 50,000 lines of Turbo Pascal code that makes up the bulk of a product still in development. I've made the conscious tradeoff not to optimize to the extreme that Mr. Greenblatt has so that I might have hope of porting some of my code to another environment should I choose to do so in the future. I've done my own "ludicrous" optimizations in my software while remaining with "pure" Pascal—ludicrous because I've spent an inordinate amount of time redesigning algorithms just to generate faster Turbo Pascal code. The algorithms are not any better off for it. This kind of boost of productivity I don't need. The fact is that if speed of generated code is the major criterion in choosing a language, Turbo Pascal is not a good choice although Version 4.0 is considerably better than 3.01 in this and all other respects. I got a long way down the path before recognizing that tight, fast generated code is not Turbo Pascal's forte, and I suspect that Mr. Greenblatt might express similar sentiments. TP4 does none of the register and subscript optimizations done by the FORTRAN II compiler for the old General Electric 225 computer, a computer I learned a lot from *25 years ago*. Nor does it eliminate common subexpressions within the same code block. Because TP3 generated slow code, Kim Kokkonen's TurboPower Software has a market for Turbo Optimizer. When a compiler does not use basic code optimization techniques common 25 years ago, it still needs improvement. A very recent Microsoft ad advertised a toll free number from which one could obtain information about a Turbo Pascal to C converter. Unfortunately, I know of no PC Pascal compiler today that has options both to compile fast *and* to generate reasonably efficient object code. In an intensely competitive PC compiler market, if I were Borland, I would be working toward releasing a Turbo Pascal "Release 4.5" within the next three months—available to current 4.0 users in diskette only form for a modest upgrade fee. Based upon what I've seen of the code generated by TP4, I would estimate that execution times of programs could be reduced in the neighborhood of 25% with reasonable optimization of register use.

Finally, I found Michael J. Young's article on Turbo C TSRs to be clear and informative. But, I'll sing what has become an old song by now. Please, since *PJ* is by and for programmers, can we have TSRs that also can be disinstalled? Mr. Young's article leaves the naive with the mistaken impression that it is perfectly all right to leave a TSR in memory until the PC is powered down. Is he whetting our appetites for his product or his book wherein he reveals all? Too many TSRs in the market take memory 'til power off do us part. Now that I've gotten the canned speech out of the way, here are other potentially useful observations about Mr. Young's article.

I don't use Turbo C, so I can only suspect that the C **keep** function is the culprit. Now that OS/2 is upon us, the old test for DOS release of 2.0 or later is insufficient. The DOS release must also be less than 10, to make sure that a TSR does not run in the OS/2 black box, where strange and mysterious things may happen with TSRs.

Mr. Young's **TsrInstall** function places a special code in a free vector in the range 60h to 67h, to indicate that the TSR has been installed. This is well and good, as long as the user of a TSR does not install another TSR that unwittingly uses the same vector that Young's TSR has chosen. This is not a sound commercial solution. Far better to identify a TSR to interrupt 28h as another process that uses it, then to use the roll-calling function of interrupt 28h to see if the TSR's process number is already assigned. Even the old standby of placing a TSR signature key at a fixed offset is less risky than using Mr. Young's method.

I find it ironic that *PC Magazine*, not often recognized as a source of information for programmers, has published several first rate TSRs in recent months, and none of the "technical" publications has. *PC Magazine*'s recent TSRs disinstall themselves on request and generally make best use of shared resources in the PC.

After I get my product out and in the marketplace, I would be willing to contribute a short piece to *PJ* regarding TSRs. It would consist simply of a MASM TSR that does nothing at all—except to disinstall itself when requested. I know of at least three variations for removing a TSR from memory. In the meantime, if someone else makes a similar contribution, many readers of *PJ* will be better off for it.

Ben Myers
Harvard, MA

Michael Young Replies

I agree with Mr. Myers that there must be some means for de-installing a TSR besides rebooting the machine! However, as I should have mentioned in the article, I lost all interest in devising mechanisms for de-installing my TSRs when I discovered the excellent shareware utilities MARK and RELEASE from TurboPower Software. A TSR that provides a de-installation function trusts that the user will use this function only if the TSR is the *last one* installed in memory. Otherwise, a large gap will be left in the chain of allocated memory blocks maintained by DOS, and there is also a danger that subsequently installed TSRs will chain control back to the released block. The RELEASE utility, however, safely de-installs all TSRs installed since MARK was run; it automatically releases the TSRs in the correct order, returns all memory to DOS, and cleans up the interrupt vector table. If I were distributing a commercial TSR program, I would include similar separate utilities rather than allow the TSR to de-install itself.

Mr. Myers' second point—that you should test that the DOS version is greater than or equal to 2 but less than 10—is valid assuming that you do not want to install the TSR in the real mode box of OS/2 (Version 1.0 of OS/2 returns a version number of 10.0 to distinguish it from MS-DOS). However, I have tested the Turbo C TSR in the OS/2 real mode screen group, and it works without problem. The OS/2 real mode, although admittedly a precarious environment with the potential for crashing the entire system, does support interrupts 28h and 21h and services 34h, 50h, and 51h.

Regarding Mr. Myers' last point, I am not familiar with the interrupt 28h roll-calling function. It sounds like a good solution to the problem of preventing TSR re-installation, and I hope Mr. Myers will explain the details of this function in a letter or article.

Michael Young
Mill Valley, CA 94942

The Fortunes of OS/2 and PS/2

Out of slightly more than 100 systems for MS-DOS we have shipped since early September, only two (2) have gone out on 3.5-inch disks! We received an additional inquiry from someone who converted to 5.25-inch disk rather than pay a $5.00 charge. Nobody has asked about OS/2.

Now, 2–3% is not a very high penetration rate for the PS/2 series, and you may ask what types of customers we have. We offer a neural net package that sells primarily to university professors, R&D labs, and consultants. We have a sophisticated customer base involved with current research.

Needless to say, I cannot justify developing for the PS/2 series based on demand so far. Are others seeing similar figures?

Dennis Reinhardt
President
DAIR Computer Systems
Palo Alto, CA ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Software Tools

❑ **The DATE UTILITIES** let you copy, erase, or display files based on their name **and** date. You can also change a file's time stamp. All utilities support wildcards for file names and let you limit the files chosen to those created **before, on,** or **after** a specified date. The package includes DCOPY, DERASE, DDIR, CDATE, and three other general purpose utilities. Version 1.5 costs $47.50.

❑ **PROINDEX** is a professional document indexing package. It works with all word processed text to create an index in any style. PROINDEX compares your text file to a **list** of keywords or phrases. These keywords **or** phrases can be up to 70 characters long! No limit on the size of the file to index or the number of references. Version 2.24 costs $89.95.

❑ **Soft Fonts** for the HP LaserJet and compatible printers. We offer **inexpensive** Helv, Roman, and Script font families. We have Roman fixed width, Bar code 39, Title fonts, Symbol sets, and Credit cards. All font families are $50 or less! Printer drivers for most word processors cost only $15 per font family! Custom fonts & a download utility are also available.

All tools work on the IBM PC or any compatible, MS-DOS 2.1 & up. No charge for shipping in the U.S. Thirty day money back guaranty!

ECI P.O. Box 61, Wasco, Illinois 60183

VISA 312-377-3520 MC

Number **70** on the Reader Service Card

DEVELOPMENT TOOLS FOR C

- A COLLECTION OF "TOOL BOXES" FOR THE DEVELOPMENT OF POWERFUL C PROGRAMS
- DESIGNED TO BE USED BY BOTH NEW AND EXPERIENCED C PROGRAMMERS
- EACH TOOL BOX IS FULLY DOCUMENTED -- WITH EXAMPLES

SYSTEM TOOL BOX
- ACCESS THE POWER OF YOUR PC
- KEYBOARD & SCREEN MANAGEMENT
- GRAPHICS & MUCH MORE
- 184 PAGE USER'S GUIDE

LIB + SRC $35

UTILITY TOOL BOX
- GENERAL PURPOSE FILTERS
- UTILITY PROGRAMS
- GENERAL PURPOSE FUNCTIONS
- 86 PAGE USER'S GUIDE

LIB + SRC $25

WINDOW TOOL BOX
- COMPLETE WINDOW CONTROL
- BUILD WINDOW DATABASES
- FULL ERROR REPORTING
- 84 PAGE USER'S GUIDE

LIB $45 -- SRC $125

MENU TOOL BOX
- VERTICAL/HORIZONTAL MENUS
- CREATE MENU DATABASES
- FULL ERROR REPORTING
- 78 PAGE USER'S GUIDE

LIB $35 -- SRC $75

- HARDWARE -- IBM PC, TI PRO OR TANDY 1000
- COMPILER -- MICROSOFT™ C (3.00 & LATER)

- CALL OR WRITE FOR FREE INFO & DEMO PROGRAM
- TECHNICAL SUPPORT IS UNLIMITED & FREE
- NO ROYALTIES

FLEMING SOFTWARE
5741 Wagonwheel Trail
Castle Rock, CO 80104
(303) 688 - 3584

Number **118** on the Reader Service Card

Software Tools
For Programmers & Non-Programmers

★ NEW ★ VERSION **Opt-Tech Sort**™ ★ NEW ★ VERSION

Opt-Tech Sort is a high performance Sort/Merge/Select utility. It can read, sort and write a file faster than most programs can read the data. Example: 1,000 records of 80 bytes can be read sorted and a new file written in less than 9 seconds (IBM XT). Opt-Tech Sort can be used as a stand-alone program or Called as a subroutine to over 30 programming languages.

Numerous features are included. A partial list of features includes: The ability to process files of any size. Many filetypes are supported including Sequential, Random, delimited, Btrieve, dBASEII & III and many others. Up to 30 key fields can be specified over 16 different types of data are supported. Powerful record selection capability allows you to specify which records are to be included on your output. Record reformatting allows you to change the structure of your output record and to output special fields such as record numbers for use as indexes. **Newest Features: Faster than ever! LIM EMS support. User exits to pass or receive records. Less disk space required and much more!** MS-DOS $149. Upgrade $35.

On-Line Help™

On-Line Help allows you to easily add Help Windows to all your programs. On-Line Help is actually two help packages in one. You get BOTH Resident (pop-up) and Callable help systems.

The resident version allows you to add help to any system. Your Help system is activated when the "Hot Keys" that you specify are pressed. You can then chain between help windows in any manner you desire. Upon clearing the help windows you are right back into what ever program you left.

The callable version allows you to easily display help windows from your programs. A simple Call to the help system makes the window appear. On-Line Help is callable from over 20 different languages.

You have full control over the help window content, size, color and location. MS-DOS $149. Demo $10.

Scroll & Recall™

Scroll & Recall is a resident screen and keyboard enhancement. It allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 screens of data can be recalled or written to a disk file (great for documenting systems operations). Also allows you to easily recall and edit previously entered DOS commands without retyping.

Scroll & Recall is very easy to use. Its a resident utility that's always there when you need it. MS-DOS $69.

Visa, M/C, AMEX, Check, Money Order, COD or Purchase Orders accepted

To order or to receive additional information just call and receive immediate highly qualified attention!

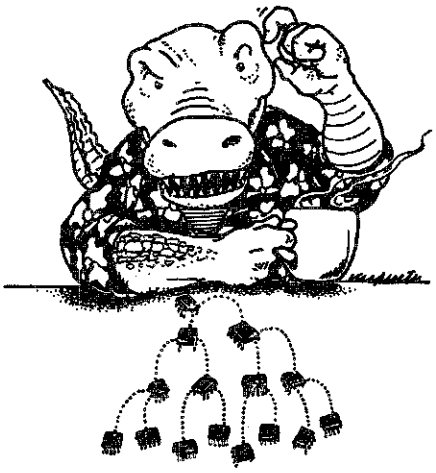
Opt-Tech Data Processing
P.O. Box 678 / Zephyr Cove, NV 89448
(702) 588-3737

Number **16** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Memory Trends

by
Hal W. Hardenbergh



In the last issue, I discussed the clear historic doubling of microprocessor CPU performance every two years. What are the corresponding trends in memory size and speed, and how will these trends affect future personal computers?

Memory Size Trends

In 1960, Texas Instruments introduced the type 502 integrated flip-flop for $400. So what? So a flip-flop is a one-bit static RAM! This was the first integrated semiconductor memory chip.

In 1971, Intel introduced the first 1K dynamic RAM, the 1103. It got cheaper later. In 1976, 4K dynamic RAMs were cheap, and Mostek introduced its 16K version with multiplexed address lines. RAS and CAS became new buzzwords. In late 1987, 1 Meg DRAMs would have been getting cheap except for stupid U.S. government interference, and Toshiba started shipping engineering samples of its 4 Meg DRAM. Historically, successive generations of DRAMs have had 4 times the capacity of the preceding generation.

For the sake of argument, let's say that 1988 is the year in which 4M DRAMs are sampled. That means we have gone from 1K to 4M in 17 years. If that were 18 years, we could say that DRAM memory quadruples in capacity every three years—and it's close enough that I'm gonna stick with that three-year figure in this article. (Cost-effective volume production normally trails engineering samples by about two years.)

Nothing doubles forever. How much longer can we expect DRAM capacity to double at the current rate?

The 4 Meg generation is already here: That's half a megabyte on *one* chip! It appears that there are no insuperable technical obstacles to prevent the appearance of the 16 Meg generation on schedule three years hence. Can we depend on the 64 Meg to appear in six years? The 256 Meg in nine years?

The current 1 Megs are being made with roughly 1.2 micron design rules. GCA is already shipping eximer-laser steppers that can be used to make 0.5 micron devices, or even 0.35 microns with some resist improvements (although they're too slow for production, they're fine for development work). So 16 Megs would appear to be in the bag. Beyond that? Good question.

Remember, there are *always* problems with the next generation(s). When all of the early 64K DRAMs proved unreliable, it was thought that 256Ks might be the end of the line . . .

Recent papers from IBM and AT&T discuss inherent leakage mechanisms in 4 Meg DRAMs. The IBM paper concentrated on "trench capacitor" DRAM structures, but the AT&T paper discussed essentially identical leakage mechanisms in normal (non-trench) capacitors. Right now, both NEC and TI are reported to be having yield problems with their 1 Meg DRAMs, and both use trench capacitors. That's one thing to worry about.

Fundamental limits? A micron is 3,000 silicon atoms wide. If we need 88% accurate alignment of two-dimensional chip areas (the two "plates" of the DRAM capacitor, for instance), we have to have 6% alignment accuracy in each of two dimensions. So the alignment accuracy is 180 atoms at 1 micron—and fewer than 100 atoms at 0.5 microns!

Another limit: Dielectric stress, measured in volts/unit length, is now becoming a problem. 5 volts per micron is 5 million volts per meter. IBM has long favored a two-step reduction of the supply voltage used by DRAMs, first to 3.3 volts and then to 2.0 volts. In fact, IBM's 4 Meg design uses 3.3 volts. The problem is that us TTL logic designers are hooked on 5 volts and the first company to change to a non-standard voltage will face enormous market resistance. Since IBM sells only to itself, it can afford to make this move.

A *serious* non-technical limitation is that the cost of making wafer fabs is climbing even faster than DRAM capacity! It takes a Class 10 clean room to produce 1 micron design-rule chips. What will 0.5 micron fabs cost? 0.25 micron fabs? Three quadruplings would take us down to 0.1 micron design rules. DRAM chip usage is not increasing all that fast, so there is no unlimited market to pay for ultra-expensive wafer fabs. We just might run out of money before we run out of atomic resolution or insulation . . .

As long as the next generation of DRAM costs the same as the last generation, there will never be an end to the quest for more and more memory. But there is a positive feedback mechanism: If prices rise, folks might be satisfied with fewer chips with the same memory capacity using X4 instead of X1 bit widths. If that happens, usage will drop, which means prices will rise further, which means . . .

Somebody might develop something that is to the planar silicon DRAM as the transistor was to the vacuum tube, but you can't predict that sort of thing. IC production techniques haven't changed since Bob Noyce developed the planar process and oxide passivation 29 years ago. My guess is that this "doubling forever" will come to an end before the end of this century due to increasing production costs. What's your guess?

A MIPS/Megabyte Imbalance

In the last issue, I pointed out that microprocessors have been around for about 17 years and that CPU performance historically doubles every two years. If memory capacity is quadrupling every three years, it is (in effect) doubling every one-and-a-half years. And that means that DRAM memory capacity is increasing faster than CPU speed. In six years, CPUs will get 8 times faster and memories will get 16 times larger.

Therefore, in six years memory capacity per "CPU smart" doubles. Both DRAMs and microprocessors have been around for about 17 years, which means that we now have 8

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

times as much memory capacity per "CPU smart" as we did when microprocessors were invented. Gene Amdahl thinks that CPU and memory are balanced when each megabyte of RAM is serviced by one MIP of CPU performance, which means that today's high-end 4 MIPS personal computers should have 4 Megs of DRAM.

The 8MHz 68000 is 13 times faster than a 1MHz 6502 on average. The difference is greater than that for math work, less for text work. An article in the Jun '82 *Computer Architecture News* suggests that a 16MHz 68000 has about the same integer CPU performance as the VAX 11/780. That puts the original Macintosh at about 0.4 MIPS and theApple II at .04 MIPS.

1977 Apple II	.04 MIPS	16K (.016 Meg)	2.5 MIPS/Meg
1982 Macintosh	0.4 MIPS	128K (.128 Meg)	3.1 MIPS/Meg
1987 386 PCs	4.0 MIPS	4 Megs	1.0 MIPS/Meg
1987 MegaST	0.5 MIPS	4 Megs	0.12 MIPS/Meg

We've left behind the early era of personal computing, when all, repeat all, personal computers had a severe memory shortage, and we're entering the era of excess memory. That's why you're beginning to see "waste" usage of memory—ramdisks, memory-resident parasitic applications, etc. 4 megabytes of memory in a 0.5 MIPS machine is ridiculous except as a ramdisk.

Memory Speed Trends, or the Von Neumann Bottleneck

CPU performance is fundamentally limited by the memory bus-bandwidth. CPU performance can double every two years only if the memory bus-bandwidth *also* doubles every two years. (Why hasn't this been a problem up to now? In the old days—6502, 8088—memory bus-bandwidth was greater than CPU performance. CPUs have just caught up.)

Memory bus-bandwidth is the product of memory speed times the bus width. We went from 8-bit buses in 1977 to 16-bit buses in 1982 to 32-bit buses in 1987, so bus width is doubling every five years. The bad news is that memory speed has been increasing slowly—by my calculations, DRAM speed doubles every seven years! Thus, memory bus-bandwidth is doubling every three years at best, much more slowly than CPU performance. We are going to have a *serious* shortage of memory performance by the end of the century. The emphasis of CPU design is shortly going to turn from the instruction set to the memory interface.

I'm a hardware guy; most of you are software folks. Here is what a hardware guy thinks are the best ways to speed up the memory interface, assuming cost is no object:

- doubled bus width
- high-speed static RAM memory systems
- cache(s)
- interleaved memory
- burst mode memory reads
- static CAS, page, or nibble mode DRAM

Doubling the bus width gains an instant doubling of the memory bus-bandwidth but requires a new CPU design. Getting 64 bits of I/O on a microprocessor chip ain't easy. But the benefit is so obvious, and you are going to see 64-bit micros sooner than you think.

Because MS-DOS programs are limited to 640K, some 386 vendors are shipping systems using high-speed static RAMs for main memory. This is a temporary solution, viable only because of that 640K limit. (No solution is perfect. You might want to use a cache with your high-speed static RAM memory to free up some memory cycles for DMA.)

For a given bus width, using high-speed cache memory is the best way to interface (relatively) slow DRAM to a high speed CPU. It's also expensive—an external cache system will cost more than the CPU. The Motorola 68020 has an internal 1/4K instruction cache; the 68030 has both an internal 1/4K instruction *and* a data cache. Because these caches are part of the CPU chip, they are in effect free. All other things being equal, a small cache(s) is better than no cache, and a large cache is better than a small cache.

Interleaving the memory helps. Mainframes commonly use a 4X interleave; the Intel 386 motherboard uses a 2X interleave. Interleaving requires extra circuitry somewhere. It can be in the CPU chip itself, in an extra block of logic chips between the memory and the CPU—or it can be in the memory itself! (Video DRAM, originally designed for another purpose, can provide, in effect, internal interleaving when supported by appropriate CPU logic—as the AMD 29000 RISC chip does!) A 2X interleave lets you get 50% of your reads using zero wait states, and a 4X interleave 75%.

> The AMD 29000 executes most instructions in 40 nsec (25MHz). What kind of memory are you going to use to keep that CPU busy? How are you going to sneak in some I/O cycles? DMA I/O? Almost any real-world memory system design is going to significantly limit the CPU's performance.

Burst mode means reading several words of memory very quickly, more quickly than those same words could be read one at a time using separate read cycles. The idea is to sidestep the slow DRAM read cycle time. A burst mode read gets words from memory using static CAS, nibble, or page mode DRAM (or even interleaved memory) and sends that burst to a cache or to an instruction prefetch queue. The presence of a cache implies burst mode reads.

A typical DRAM might have a cycle time of 200 nsec, a RAS/CAS access time of 120 nsec, and a nibble or page mode access of 50 nsec for subsequent nibble fetches. Burst mode always has to pay (at least) the cycle time overhead for the first word, but subsequent words have significantly less overhead. For the figures just mentioned, it would require 800 nsec to fetch 4 words without burst mode, but only 350 nsec with burst mode.

Most modern DRAM chips have some hardware mechanism to support burst mode reads. The most common is nibble or page mode, but some static CAS chips are available. Nibble mode is a method of getting 4 consecutive bits from a DRAM; page mode permits bypassing the precharge and RAS delays, which means that memory with a narrow range of addresses—512 bytes, for instance—can be accessed with only a CAS setup delay. Static CAS mode is the same as page mode except that no CAS strobe is needed; the 512 bytes (or so) is addressed as static RAM. AMD's use of video DRAM further improves on this idea; too bad video RAMs are bigger and more expensive than regular DRAM. (It is tough to say whether the use of video DRAM is page mode or interleaved memory, with the interleaving done inside the memory chip. The end result is pretty much the same.)

The very best solutions are combinations of the above. Caches are almost always used with some form of burst-mode DRAM and may also work with interleaved memory for a further speed improvement.

We can't ignore dollar tradeoffs; your cheap 386 clones are not going to use caches and so will run a lot more slowly than the high-priced units with caches. How fast is an Ajax 16MHz 386 clone? Not enough info, chum: Tell me about the details of the memory system, and I might be able to venture an opinion.

Continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

And that's the entire point. Right now, *today*, the CPU and clock rate no longer determine the speed of a microcomputer. And the problem is going to get four times worse by the end of this century because the CPU speed trend has a steeper slope than the memory bus-bandwidth trend. *Memory bus-bandwidth is now the scarcest microcomputer system resource!*

And *that* leads to another conclusion: The CPUs that are going to win the performance race are the ones that make the most effective use of the scarcest CPU resource—the CISC machines. That's because a single CISC instruction does, on average, more useful work than a single RISC instruction. *RISC machines inherently require more memory bus-bandwidth to get the same job done!*

Hmm. I said that last issue, didn't I?

A Prediction

In 1993, a high-end personal computer will have a 64-bit memory bus and will use 16 Meg DRAMs, organized 16 Meg x 1 bit. That means that a *minimum* memory configuration will be 128 megabytes! That'll be enough memory to run the absurdly inefficient operating system that'll be promised for the machine by 1996. The 30 MIPS CPU (aided by a big, fast cache memory) will be able to run that operating system almost as fast as 1977's Apple II.

Cheap clones in 1993 will still use the old 4 Meg DRAMs and a 32-bit bus. The advertised configuration will have a mere 16 megabytes of DRAM, but most folk will add another row of DRAM for 32 Megs. The 8 MIPS 486 CPU (with its small internal instruction and data caches) will be widely regarded as obsolescent, but it will run OS/2 satisfactorally, and there will be lots of application software.

Miscellaneae

Japan's NTT has a prototype 16 Meg DRAM. Toshiba's 4 Meg, which is now being sampled, uses 0.8 micron design rules and measures just over 1/4 inch by 1/2 inch (6.54mm x 14.9mm). Successive generations of DRAMs are 25% faster and have 40% more chip area than the previous generation.

Shipments of 256K DRAMs peaked in the first quarter of '88, and production of 1 Megs is building rapidly. First-quarter figures for Japanese shipments are 164 million 256Ks and 31 million 1 Megs (MITI's projections are usually exact). Note that 1 Meg shipments have almost caught 256Ks on a capacity basis.

If you were wondering why Intel wants to keep the 386 sole-source, you might like to know that a financial analyst has estimated that 78% of Intel's '88 profits will come from 386 family sales.

When Thorn-EMI tried to sell Inmos, it found several "buyers" who all wanted Thorn to *pay* them for accepting Inmos and its wonderful Transputer! Thorn is now looking for a partner to help keep Inmos afloat. One of the problems is that Inmos' 2 micron fabs are now outdated. The cost of new fabs does not immediately affect the profit statement because the cost is balanced in the books by the fabs' asset value. Translation: Inmos might actually report a small profit in '88 while eating *tons* of *dinero*. A lot of semiconductor firms are in this same boat: reporting profits while eating money . . .

The superficial coverage of cache memory in this article was forced by space limitations. In fact, I don't even know of a *book* that is up-to-date and fully covers cache memory.

For instance: The internal caches in the 68030 are addressed using *virtual*, not direct, addresses. That means that bus-watching cannot be implemented in a multiprocessor environment. This will impact .001% of sales at the high end, and that worries Motorola a lot. Motorola has never cared for the 99.999% of 680x0 sales at the bottom end. The Nat Semi 32532, which also has dual internal caches, addresses them directly, which means that a bus-watching multiprocessor environment *is* possible. (The problem is that instructions and data reside in internal cache *and* in global memory. It would be very nice if they were the same for all of the processors.)

Hitachi's G32-200 microprocessor uses 64-bit internal data paths and a 32-bit external bus, analogous to 1980's 68000 with its 32-bit internal bus and 16-bit external bus. You are going to see true 64-bit microprocessors *soon*. ♦

If you can find better sort/merge software, buy it!

Sortex:® A flexible, general purpose sort/merge facility which provides you with the ultimate in performance and reliability

- Sortex can be run from the DOS prompt, batch file, or applications written in Assembler, BASIC, COBOL, FORTRAN, Pascal, or C.
- Sortex can accept your requirements through the full screen menu/panel facility, control file, or command line.
- Sortex can process most of the commonly used file and data types.
- Sortex can produce output files with key, address, index, tag, or full records.
- Sortex imposes no practical limitations. Files with two billion characters, records & fields with thirty two thousand characters each, and unlimited number of keys can be handled by Sortex.
- Sortex allows records to be chosen selectively for processing.
- sortex can sort your records in ascending/descending order on any number of keys.
- Sortex can merge your sorted record sequences into one sequence on any number of keys.
- Sortex preserves the input order of records on the output file if all their keys are identical.
- Sortex allows keys to overlap, or be contained within other keys. They need not be contiguous.
- Sortex can process files that contain standard & nonstandard field delimiters, record separators, & file terminators.

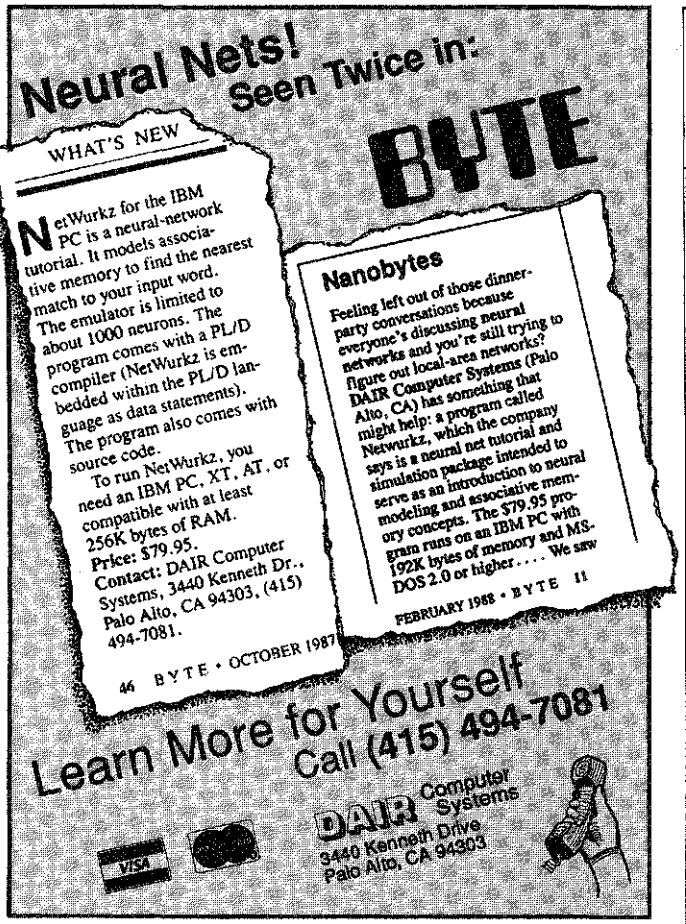
30-Day Money-Back Guarantee Only $99.95

Systemat CORPORATION

1915 W. Orangewood Avenue
Orange, California 92668
Telephone: 714 978 0480 Telex: 910 250 0674

Number 171 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED


Neural Nets!
Seen Twice in:
BYTE
WHAT'S NEW
NetWurkz for the IBM PC is a neural-network tutorial. It models associative memory to find the nearest match to your input word. The emulator is limited to about 1000 neurons. The program comes with a PL/D compiler (NetWurkz is embedded within the PL/D language as data statements). The program also comes with source code.
To run NetWurkz, you need an IBM PC, XT, AT, or compatible with at least 256K bytes of RAM.
Price: $79.95.
Contact: DAIR Computer Systems, 3440 Kenneth Dr., Palo Alto, CA 94303, (415) 494-7081.
46 BYTE • OCTOBER 1987
Nanobytes
Feeling left out of those dinner-party conversations because everyone's discussing neural networks and you're still trying to figure out local-area networks? DAIR Computer Systems (Palo Alto, CA) has something that might help: a program called Netwurkz, which the company says is a neural net tutorial and simulation package intended to serve as an introduction to neural modeling and associative memory concepts. The $79.95 program runs on an IBM PC with 192K bytes of memory and MS-DOS 2.0 or higher.... We saw
FEBRUARY 1988 • BYTE 11
Learn More for Yourself
Call (415) 494-7081
DAIR Computer Systems
3440 Kenneth Drive
Palo Alto, CA 94303
VISA

Number 5 on the Reader Service Card

Systems Tools for Turbo C

A comprehensive function library by the author of *Systems Programming in Turbo C* (Sybex,1988)

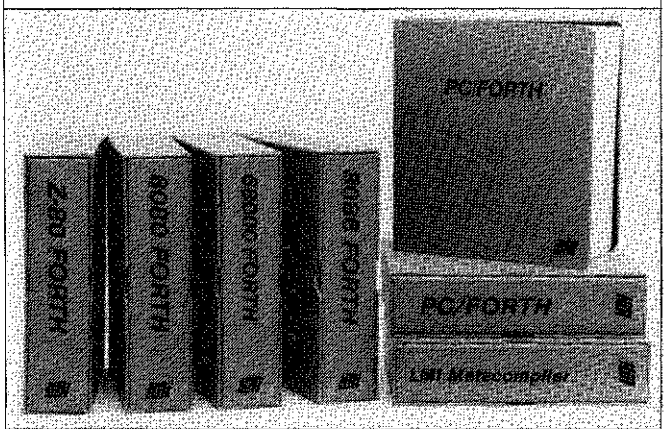
o new o written specifically for Turbo C o save time developing high performance applications o compatible with the PC, AT, & PS/2 o convert a Turbo C program to a TSR activated by a hotkey o create screens, windows, & menus with an interactive designer o rapidly display screens & strings o read data from specific fields on the screen o display, stack, & move windows o control the cursor, video modes, & display attributes o write data to a large virtual screen o scroll & view a virtual screen through a window o manage the keyboard o manage the printer o swap printer ports o format printed reports & forms o access the DOS print queue from Turbo C o manage directory names, volume labels, & file dates & times o open up to 255 file handles simultaneously o control-break & critical-error handlers o a complete device independent graphics package o an LIM expanded memory interface o over 100 functions o 2 diskette set o ready-to-link library files for all memory models o complete commented source code included o printed manual o many example programs o no royalties o $39.50 introductory price o

Send check or money order for $39.50 plus $2.50 shipping and handling (California residents please add 6% sales tax). Your Turbo C toolset will be shipped immediately.

Michael J. Young, Software Engineering
P.O. Box 5068
Mill Valley, CA 94942
415/383-5354

Number 168 on the Reader Service Card

TOTAL CONTROL with *LMI FORTH*™



For Programming Professionals:

an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development: Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

Number 63 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Pink Noise

by
Frank D. Greco

Hissssss Admin
Strong rumors suggest that the next generation Sun Microsystems workstation will have no Unix system administration to be worried about. One of the biggest reasons Unix machines have not been popular with end users is that a Unix machine usually needs care and feeding to stay healthy; this is what's known in the Unix community as "system administration." Anyone who's installed an auto-answer modem and UUCP on a Unix system really knows what frustration means. ("Did you say I have to modify the kernel to make my Smartmodem answer a call ?") Sun plans to ameliorate Unix system administration as we (the Unix community) know it today. According to my contacts, Sun has committed many people to this problem, and the word is that the next generation user interface to many yet-to-be-announced products will use Macintosh-style front ends (Sun's NeWS will be the underlying window system). Sun isn't the only company developing these types of products. Steve Jobs' Unix gurus are also trying their best to eliminate the headaches of system administration for their NeXT workstation.

CRISP-y Critter Will Not Be Burned by the Sun
At the last Sun User Group Conference, I found out that the head developer of the AT&T Bell Laboratories CRISP RISC-based processor has left. After wrestling with Ma Bell's bureaucracy to step up the marketing of the CRISP chip for five long years, the Murray Hills researcher is now a Sun employee. The talents of this highly-touted technician will probably be employed in future SPARC development.

cp cat
Inside sources at a recent technology seminar have revealed to me that Sun is hard at work creating an Intel 80386 hardware emulator that will run on their new SPARC-based workstations. As SPARC-based implementations get faster and faster, users and programmers will be free from Intel's product timetables. Word has it that the emulation will be similar to the impressive Insignia SoftPC product that runs IBM PC XT software on non-Intel processors. The idea is not to emulate DOS, but to emulate the hardware of the PC. That way you use an actual, licensed copy of DOS (or OS/2), complete with bugs and undocumented features.

By the way, Insignia has told me that they will soon announce a PC AT emulator with EGA support for the Sun workstation.

A very long time ago (*PJ*, May/June 1985), I said that Sun Microsystems would come up with the next generation PC. I emphatically predicted that Sun would create a high-powered low-end system. With Sun's Application Binary Interface (ABI) and the programmability/aesthetics (and fun!) of NeWS, I have to say it again: Sun will capture a significant portion of the next generation PC market.

My contacts in the Valley have Sun announcing their 80386-based workstation around March or April 1988.

Remember, too, when back in *PJ*, March/April 1987, I predicted that Counterpoint would be in trouble within a year? Well, I happened to hit the nail exactly on the head. The Oriental giant Multitech has bought the assets of the beleagured workstation vendor to try to correct some of its deficiencies and cut losses. Counterpoint was reported in the industry press as a potential threat to Sun. Their reportedly scaleable System 19 workstation (which crashed everytime that I saw it) and strong financial alliances with AT&T and Kyocera made headlines in early 1986. Because of their confused strategy, high prices, and unstable software, Counterpoint was pointless. Score another one for Frank G.

Silicon Aces
If you ever get the chance, check out Sun's NeWS window system running on a high-powered Silicon Graphics workstation. The Silicon Graphics workstation graphics hardware makes low-cost graphics assist chips such as the 34010 and the 82786 look like sluggish 4.77 MHz 8088s. While you're checking out the flashy NeWS on the zippy SG workstation, also check out their realistic and networked flight simulator game. This game was written internally and pits network users against each other in real-time! Any aircraft viewed on the workstation screen is actually another workstation user on the network. A Silicon Graphics developer told me these games are so intense during after-hours dogfights that these network pilots actually don helmets and flight jackets!

Mainstream Mainframes
National Advanced Systems, who sells (Japanese) Hitachi-made mainframes, recently licensed Sun's Network File System (NFS). Another large Japanese company, Fujitsu, who owns 49.5% of IBM-mainframe-compatible Amdahl, has the first license to manufacture Sun's Scaleable Processor ARChitecture (SPARC) chip. Guess which company benefits the most? Hint: It's not NAS, Hitachi, Fujitsu or Amdahl. *Domo Arigato, Bill Joy San.*

Getting Hyper
Sun and N.V. Philips have announced their intentions to jointly develop a hypermedia authoring kit based on Compact Disk Interactive (CDI).

IBM Jumps on Sun's Bandwagon
IBM announced support of Sun's industry standard Network File System (NFS) on the PC RT running AIX. NFS allows the accessing of files among workstations on a network in a "transparent" way. File accessing semantics remain the same for any file on the network. NFS basically works by intercepting the file-oriented operating system interface calls such as **read()** and **write()** and possibly rerouting them. By "sitting below" these calls, NFS determines if a file is local or remote through a table called the standard Unix "mount table." This table is the "glue" that allows other local filesystems to be connected into one large filesystem; this is how standard Unix "mounts" filesystems. NFS just adds the ability to connect remote filesystems located somewhere on your network. If the file is found to be remote, the proper networking calls are made that retrieve it transparently to the application program. MS-DOS borrowed this concept, and its JOIN utility used it (in a very limited fashion). Try the following for a small taste of what NFS can do:

```
C:\>MKDIR \MOUNT
```

This is a temporary "mount point."

```
C:\>JOIN A: C:\MOUNT
```

"Mount" the A: drive there.

```
C:\>CD\MOUNT
```

Go to the mount point.

```
C:\>DIR
```

Files from your floppy Drive A: will be listed.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The constraints that the old-fashioned MS-DOS filesystem put on JOIN are somewhat ridiculous: You can mount only an entire disk, and the mount point must be in the root directory.

Newer research technologies have recognized the need to mount not only remote filesystems, but foreign filesystems such as MS-DOS, OS/2, VMS, optical disks, etc., as well. The Virtual File Systems (VFS) concept developed by Berkeley and Sun Microsystems solves many filesystem connectivity problems involved with mounting foreign filesystems and will soon work its way into NFS.

1300 MIPS Processor

Developers at IBM's Watson Research Center in Yorktown Heights, New York, are hard at work on a Research Parallel Processing Prototype (RP3). When this high-performance machine is fully configured, it should run at a peak MIP rating of 1300! Colleagues at New York University say that IBM has actively encouraged universities and research centers to collaborate with Big Blue on this long-term project. This monster is said to run 4.3 BSD Unix or Carnegie Mellon's impressive Mach operating system.

Roach Hotel

When I put my ear to the rail, I hear rumblings from the RT group at IBM. You can expect IBM to make a significant announcement regarding Unix and workstations within a few months. Apparently, the boys in blue have been trying to lure some Wall Street firms away from industry standard Sun workstations. From what I hear, the blue-suiters have been pointing out how powerful the proprietary RT with a proprietary version of Unix (see **signal(2)**, **socket(2)**, and VRM device drivers in the RT manuals for examples) will be when it incorporates their proprietary Micro Channel bus architecture and makes use of proprietary multi-processing communication boards. IBM customers check in to the IBM hotel, but they don't check out. The latest RT, Model 90, is not the last word (unfortunately) from IBM.

Speaking of the IBM PC/RT, corporations are starting to buy it for multi-users. Many buyers look at the PC/RT as a low-end, multi-user Unix machine from IBM. A lot of corporate MIS managers would like to use Unix in their shops—there are some really good network management, page composition, and graphics software for some Unix boxes. But no existing Unix vendor has approval from high-level management. AT&T systems usually do, but their reputation in computers is the subject of stock jokes; Sun still has (for some unknown reason) a "techie" reputation; and the feeling towards DEC is that its price/performance is not that good. Here comes the IBM PC/RT from the Armonk monster; since it's from IBM, it's already corporate-sanctioned. I have the feeling IBM may be considering the PC/RT as the System 3X replacement. Lets face it, the PC/RT is one casualty of the Unix workstation war. Sun, Apollo, DEC, HP, and possibly Apple are still fighting it out. If IBM releases some I/O coprocessors for the RT, this may add more credence to my RT/System 3X hypothesis. The elephantine IBM still has this multi-user mentality; they haven't realized that the world is going towards networked, single-user, multitasking workstations.

IBM Sticking to Magnetic

IBM announced that using semiconductor technologies it may be possible to create 3.5-inch hard disks capable of storing 10 gigabits within the next five to seven years. IBM researchers in San Jose are using cobalt-based thin film and photolithography to achieve these amazing bit densities.

When You're Hot . . .

The hottest data base company right now is Sybase, Inc. Started by Dr. Robert Epstein, one of the developers of the Britton-Lee data base hardware and Relational Technology's Ingres data base system, Sybase has captured the attention of *major* players in the industry. Microsoft, Apple, and Sun have all signed large deals with this little upstart. Sybase offers features in its data base such as a true object-oriented data base, abstract data types, and an object-based user interface. The rumors have Apple sinking 1 million dollars into Sybase; Microsoft may dish up some *dinero*, too. Apparently, Microsoft wants to counter IBM's OS/2 extended version (that is, OS/2 with an embedded data base query language) with some data base facilities of their own. Many corporations in the Northeast are carefully evaluating Sybase, and all preliminary indications are positive. An anonymous representative from Sun Microsystems pointed out to me that all Sun sales representatives are recommending either Sybase or Oracle to their clients. Other investors interested in Sybase include Hambrecht & Quist, TRW, Perkins Caufield & Byers, and there are still others. Look for Sybase to go public very soon.

Mach-intosh?

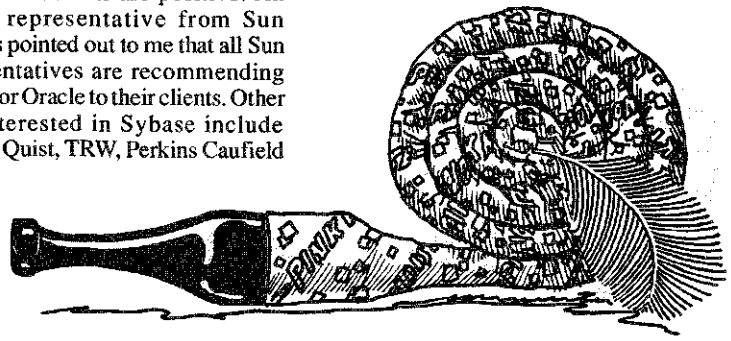
I've heard from reliable sources that Steve Jobs' NeXT workstation is running Carnegie Mellon University's Mach operating system. Apparently Jobs is keenly interested in using the power of this research operating system. Mach is binary compatible with Berkeley 4.3 Unix, maintains system-level compatibility with vanilla Unix, and has many other impressive features. Certain aspects of CMU's Mach OS are so impressive that many existing operating systems will incorporate them. I will have more information on Mach in a future column.

Rumors in some industry rags have it that the NeXT box will be announced with Unix System V.3 and X Windows. If Jobs does indeed offer X Windows after calling it "brain-damaged" in his keynote delivery at the last Usenix conference, he will have to side step many a pointed question.

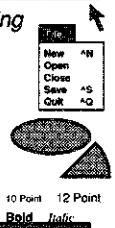
My valley contacts relayed to me that the NeXT machine will have a futuristic look similar to Jobs' Pixar machine's. The word is that Jobs has commissioned the same graphics designer, Hartmut Esslinger from West Germany, to do his thing.

Take Your pic (or Was That PICT?)

Avatar Corporation of Pittsburgh, Pennsylvania, has a software utility called MDPIC that can translate PICT files (MacDraw files, for example) into Unix **pic** format. Once the image is in **pic** format, it can be incorporated into any Unix **troff**-style document. For those of you who don't know, **pic** provides a slightly (only "slightly") easier method of drawing images onto the printed page than raw **troff** does. If **pic** files are to be printed, they must be post-processed into the Unix **troff** language. **pic** is similar in philosophy to Unix's **eqn** or **tbl** utilities. So with MDPIC, you can incorporate MacDraw illustrations directly into a **troff** document. For those of you designing corporate-level (that is, heterogeneous) electronic publishing sys-



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MetaWINDOW
Power Graphics for your PC!
PC TECH JOURNAL
"Product of the Month"
"... a technological tour de force for fast PC graphics."
NO ROYALTIES!
MetaWINDOW is ... graphics ...
NEW! - QuickWINDOW/C
All the features of MetaWINDOW for Microsoft Quick/C! - $95*
... drivers and ... window managers.
Unparalleled Performance!
MetaWINDOW provides an expanded set of graphic drawing functions, plus the added functionality and performance required for designing multi-window desktop applications.
• auto-cursor tracking
• pull-down menus
• pop-up windows
• comprehensive graphic functions
• multiple fonts


Enhanced Features!
• Display multiple bitmap or "filled-outline" fonts.
• Face fonts for bold, italic, underline or strike-out stylings.
• Full "RasterOp" transfer functions for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.
• Create pop-up menus, windows and icons.
• Supports IBM's new PS/2 VGA and MCGA graphics.
MetaWINDOW comes complete with language bindings for 20 popular C, Pascal and Fortran compilers, plus dynamic runtime support for over 50 graphics adaptors and input devices.
MetaWINDOW
Advanced Graphics Toolkit
4 disks, 3 260 page manuals - $195*
NEW! - TurboWINDOW/Pascal
All the features of MetaWINDOW for Borland Turbo Pascal Ver. 4! - $ 95*
* Plus $5.00 shipping and handling
TO ORDER CALL 1-800-332-1550
For information or in CA call 408-438-1550
METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Hermon Road
Scotts Valley, CA 95066
Number 143 on the Reader Service Card

tems that integrate Unix workstations, PCs, and Macintoshes, this is a utility that will certainly prove its worth.

Amazing Stories

I have two "hold-on-to-your-hats" reports. First, I have a friend who knows a friend (you know the kind) who has seen a three-dimensional holographic scanner. Apparently, the object is laser scanned in a fashion similar to the process of creating a hologram. This information is fed by a simple serial link to a CAD program, possibly manipulated, and subsequently drawn on the screen. Then, of course, you can use this data to drive a machine that actually creates a physical (injection-molded or layer-by-layer, for instance) prototype. Just think: scan, manipulate, product! Unfortunately, this technology is about five years away from commercial realization. But believe me, it already exists in a primitive stage in certain research labs.

The other amazing report involves high density data compression. Using newly developed algorithms based on fractal geometry, chaos theory, and pattern recognition, researchers have been able to compress 24-bit high resolution color images at nominal compression ratios of 10,000 to 1. Yes, you heard that right. And for certain types of images, compression ratios of 1,000,000 to 1 have been attained and certified. However, the compression algorithm, as expected, is heavily CPU-intensive and takes many CPU-hours (100 VAX CPU-hours) to complete. Decompression, the process of reconstructing the image, is relatively fast on the order of three complete frames per second. Plus, this algorithm is non-destructive; that is, the image is completely reconstructed with no data loss as opposed to some FFT sampling techniques. When I get more information on this one, I'll pass it on to you.

French Tickler

Does anybody remember MS-DOS 4.0? It was released around September 1986 in Paris, France, and offered unprotected multitasking. As I recall, MS-DOS 4.0 teased the hackers in America by offering a pre-emptive multitasking scheduler, overlapped I/O, interprocess communication with pipes, shared memory, signals, and semaphores. I would love to have this little sucker available on the 386 with linear memory addressing. Just add lightweight processes (ahem . . . "threads" . . . okay, Microsoft?), and it would be a VAR's dream. Most VARs would not even consider OS/2 if MS-DOS 4.0 were available in America. I mean, how many vertical market applications require (in the short term) concurrent backwards compatibility with existing software and need memory protection? Now do you realize why it wasn't released in the U.S.? Now, if Borland would come out with pre-emptive, multitasking primitives for the unprotected MS-DOS environment that would be "semantically" compatible with OS/2—**DosCreateThread()**, for example—that would be a serious short-term threat to the Microsoft Machine. Add a re-entrant equivalent of the BIOS for asynchronous disk accesses (hellloooo Phoenix?) and a *real* window system that uses network protocols (and is aesthetically pleasing), and Borland might have a substantial short term OS/2 attack. Of course, this is all my own conjecture.

Most board level integrators are introducing products based on the programmable 34010 graphics gem, not on the 82786 display coprocessor.

Technical Knock Out

Texas Instruments is enjoying a brisk business in the graphics area with sales of the TMS34010 programmable graphics processor. Expect a much faster 50 MHz version in early 1988. Apparently, TI has won the first round of low cost graphics chips over Intel and the Intel 82786. Most board level integrators are introducing products based on the programmable 34010 graphics gem, not on the 82786 display coprocessor.

Speaking of the TI 34010, Metagraphics (see *PJ*, September/October 1987) will soon announce upgrades to their impressive graphics library, MetaWindow, that will support the power and flexibility of the TI chip. By the time you read this, the upgraded library will be a reality.

You Won't Be Bored by This Board

The new PG-1281 graphics board from Matrox Electronic Systems seems to be a real hot one. Resolution on this TI 34010-based board goes up to 1280 x 1024 in 256 colors. In order to differentiate this board from the *many* 34010 boards out there, Matrox has implemented some of the TI chip's commonly used graphics routines in ASICs and has hard-wired pan, scroll, zoom, and vector generation! Matrox has stated that just by using the hard-wired vector generation, you get a four- to fivefold increase in performance over the standard 34010's already hot performance! Smokin'! A NeWS-based or X Windows-based (what's MS Windows?) workstation using this board would be hot indeed.

WYSINVG—What You See Is Not Very Good

Apple's Alan Kay and Xanadu's Ted Nelson—both visionaries—have harshly criticized WYSIWYG systems and the desktop metaphor. "Mimicking paper on the screen is not very forward thinking . . . WYSIWYG systems are just paper emulators!" says Nelson. Kay says, ". . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C Math Toolbox©

Volume I

89

essential math functions

$89.00

A productivity tool for modern C programmers

Specifically designed for use with the Microsoft QuickC® and C Optimizing Compiler® 5.x under MS DOS®

Statistics

Complex numbers

Matrices, and more

For more information, contact your local dealer, or order by mail from:

SILVER STATE SOFTWARE
1000 E. William Street, Suite 100
Carson City, NV 89701
702/885-8922

QuickC, C Optimizing Compiler, and MS DOS are registered trademarks of the Microsoft Corp.

Number **186** on the Reader Service Card

For those who need a

FAST C COMPILER with 8087 & software floating point and a full DOS 3 compatible STDIO library

FAST 8086 ASSEMBLER that is stand-alone, or inline with no separate assembly step

FAST LINKER that makes stand-alone .EXE files with both memory and disk-based overlays

SOURCE DEBUGGER for both C and Assembler

FULL SCREEN EDITOR that edits 2 files with no size limits & can invoke the Compiler & Linker

GRAPHICS package with line and text functions

DeSmet DC88

We've combined our popular DeSmet C88 Development Package with the DeSmet D88 Debugger and our Graphics Package. Library source code is available separately.

$99

C Ware

P.O. Box 428, Paso Robles, CA 93447
Phone:(805)239-4620 Telex: 358185 BBS: (805)239-4834
We accept VISA, MC & AMEX. Call now and we'll ship today
945 Spring St #14, Paso Robles, CA 93446

Number **38** on the Reader Service Card

"If you need source code, make sure your wallet is wide open or get VITAMIN C."
- Computer Language, June, 1987

Vitamin C

It's good for your system!

- Professional C function library
- 30 day money back guarantee
- Multiple bullet proof windows
- Easy full screen data entry
- Unlimited data validation
- Context sensitive help manager
- Menus like Lotus and Mac
- Programmable keyboard handler
- Text editor routines
- No royalties or runtime fees
- Library source included FREE
- Free technical support
- Free BBS at (214)418-0059
- Supports all major compilers including Microsoft 5.0
- VCScreen code generator too!
- UNIX version avaialable

Fast, flexible, versatile, reliable. Vitamin C delivers the vital combination that software professionals demand. Highly efficient, professionally crafted C code provides lightning fast displays required by today's window intensive programs.

High level functions provide maximum productivity and require little supporting code. Plus, Vitamin C's versatile, open ended design is full of hooks so you can intercept and plug-in special handlers to customize or add features to most routines.

VCScreen, our screen painter / code generator speeds your development even more! Simply draw your input forms using our interactive design editor and generate perfect C source code ready to compile and link with the Vitamin C library.

Vitamin C $225
Includes all source code FREE! For IBM PC, XT, AT, PS/2 and true compatibles. Specify compiler when ordering.

VCScreen $99.95
For IBM PC, XT, AT, PS/2 and true compatibles. Requires Vitamin C.

ORDER NOW!

We ship UPS. Please include $3 for ground, $6 for 2-day air, $20 for overnight, or $30 if outside the U.S. Texas residents add 7 1/4 % sales tax. All funds MUST be in U.S. dollars drawn on a U.S. bank. Visa & MasterCard accepted.

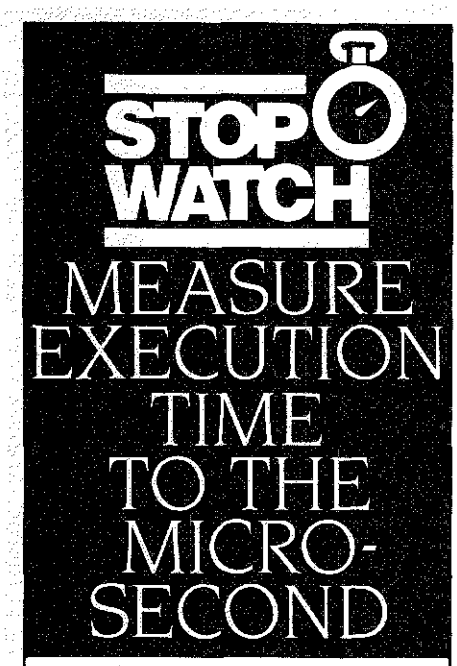
(214)416-6447

creative PROGRAMMING
Box 112097 • Carrollton, Texas 75011

Windows • Data Entry • Menus • Help • Text Editing
Plus... All Source Code FREE!

Contact this Advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



■ ***STOPWATCH™ . . . It Measures Program Time with Microsecond Resolution.*** The new STOPWATCH Program measures total and component times of programs executed on IBM PCs, XTs, ATs, and compatible units to within ± 5 microseconds.

■ ***STOPWATCH . . . It's Easy to Use.*** With STOPWATCH there's no need to modify your program for testing. Simply key ⟨ctl⟩-M instead of ⟨enter⟩ anytime you want to measure execution time to the next display output. (⟨ctl⟩-M and ⟨enter⟩ are equivalent *until* you install STOPWATCH.)

■ ***STOPWATCH . . . Use It To Time New Programs.*** Use STOPWATCH with DEBUG, SYMDEB or CODEVIEW to measure execution time from "GO" to Breakpoint. And, STOPWATCH can be used to time Lotus 1-2-3 Macros, dBASE III Procedures, BASIC Programs, or any object program that you have developed.

■ ***STOPWATCH . . . Use It To Evaluate Your Present System.*** Add STOPWATCH to your present system and measure component times of a program to quickly determine whether a new processor, or faster disk, is the key to improving response time for your application.

■ ***STOPWATCH . . . Unparalleled Accuracy At An Affordable $59.95.*** Included are demo looping programs to demonstrate STOPWATCH's timing ability, the STOPWATCH Program, and a user's manual with easy-to-follow instructions to get you started quickly. Now . . . Microsecond resolution timing — only $59.95! Put your program to the test today with STOPWATCH!

Now Microsecond Resolution Timing Only $59.95 + 3.50 S&H + 6% NJ Tax. MC/VISA accepted

US 800-345-0167
NJ 201-228-7623

CRTS
CUSTOM REAL-TIME SOFTWARE
PO Box 1106, West Caldwell, NJ, 07007

IBM is a registered trademark of International Business Machines Corp.
CodeView is a trademark of Microsoft Corp.
Lotus and 1-2-3 are registered trademarks of Lotus Development Corp.
dBase III is a trademark of Ashton-Tate.
*STOPWATCH is a Trademark of Custom Real-Time Software Inc.

Number 124 on the Reader Service Card

paper is really a 500-year-old embalming fluid for ideas."

The Tale of the Tail-less

At the last Comdex, a new mouse that uses infra-red technology to communicate with your PC was announced. The Torrington Company (203/482-9511) showed their "tail-less" 3-button mouse that sends signals to a receiver connected to the PC. Fully compatible with Mouse Systems' and Microsoft mouses, it lists for $229.

Also at the last Comdex: a long line in the QMS booth, where they were handing out samples of the new color PostScript printer. The three-pass printer (yes, the paper literally goes through the printer three times) uses special thermal paper and should cost in the neighborhood of $20,000 to $25,000. In my opinion, this is way too expensive, especially for a printer that needs specially treated paper.

Representatives of Tektronix I talked to at Comdex told me their $10,000 ($8,000 base prince + $2,000 for necessary RAM memory) model 4693D thermal-wax, full-color printer would soon support the PostScript language.

At the Howtek booth, I happened to talk to *the* developer of their Pixelmaster full color (256,000 different colors) printer that uses ordinary paper. He boasted that he was just finishing a PostScript driver that would soon be ready for the Pixelmaster and told me that the 240 dpi Pixelmaster is targeted for the $5,000 price point. Now *that* would be amazing!

Another interesting announcement at the last Comdex was Printware's (612/456-1400) 1200 x 600 dpi laser printer that uses a PostScript clone (called Printscript) and Bitstream fonts.

Seikosha Puts the Pedal to the Metal

Seikosha America, Inc. (800/422-7768) recently announced an 800 cps, 18-pin matrix printer. Both serial and parallel interfaces as well as Epson/IBM Proprinter emulation are standard. Near-letter-quality (I hate that phrase!) speed comes in around 200 cps.

Frank Talk

How about some credit, Microsoft? Included with the latest Microsoft assembler and C compiler packages is a little utility (called **errout**) that allows the developer to redirect the standard error to a logical file. I published my utility (called **err**) to do the same thing in January 1987 (see *PJ*, Volume 5.1 for the C language source)—after Microsoft had told everyone that standard error was non-redirectable (I can quote from at least two Microsoft Press books). How about some thanks, guys! Now do all those software developers out there a favor—incorporate this capability into your line-oriented shell (COMMAND.COM or CMD.EXE)!

Mail Messages

Since several people have requested System V and BSD Unix versions of my timer macros (*PJ*, Volume 6.1, "Time is on My Side . . . Yes It Is"), Listing 1 is a modification to the TIMER.H file that should help you out. If you are using System V, compile with the SYSV macro defined. Compile with the BSD macro defined if you're using Berkeley-based Unix systems. Remember that when you time things under a multitasking operating system, your results depend on what other tasks such as networking or printer "daemons," the Unix scheduler or the "update" program may be executing in the background. So under a multitasking operating system such as Unix, your timing results are only a close approximation of reality.

I didn't realize how interested you Redirections readers were in my timing routines/macros. A number of developers either called or wrote regarding them. Also, since my macro and subsequent program were limited in timing granularity to about 1/18 of a second (one clock tick), several readers, Phil Pistone of Park Ridge, Illinois, and C. Brockley of Warrensville Heights, Ohio, among them, wanted to know if it's possible to get higher resolution timing. It certainly is; however, the timer chip on the PC (either the 8253 or the 8254) must be programmed to reflect more ticks per second.

I've included some code (Listing 2) that brings the timing granularity up to one millisecond, that is, 1000 clock ticks per second. You can time events up to about 25 minutes (there are 0x001800B0L maximum 1/18 second clock ticks per day, which when divided by 1000 clock ticks per second, is approximately 25 minutes) in duration. If you want to time longer events, you must write your own timer interrupt to keep proper time since the PC's timer interrupt routine thinks clock ticks are still occurring at 18.2 times per second. Also notice that your timing results will be slightly longer than before because clock interrupts happen 1000 times per second, not 18.2 times per second. Subsequently, the servicing of the timer interrupt will soak up more valuable CPU time.

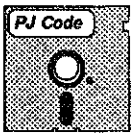
By the way, PC clock ticks occur at 18.2 times per second because the 8253/8254's clock rate is 1.19318 MHz, which when divided by the maximum clock ticks that can be counted in a 16-bit PC word results in 18.20648; that is, 1.19318 MHz divided by 65536 is approximately 18.2.♦

Frank D. Greco is an independent consultant who furnishes industry analysis, systems consulting, and technological forecasting for Fortune 500 companies. Readers can contact Frank at the following address:

Frank D. Greco
c/o Crossroads Technologies, Inc.
P.O. 530
Fanwood, New Jersey 07023
(201)-754-7820

Code follows.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



All code in this issue is available on 360K, MS-DOS disk, $10.00 postpaid. ORDER BY PHONE — VISA/MC orders call (503) 747-0800. Mail order information on page 72.

Code From "Pink Noise" by Frank Greco — 1

Listing 1

```
/*
 * TIMER.H      --  Header file for timing programs
 *
 * When including and compiling with your programs:
 *
 * System V
 *              $ cc -DSYSV program.c ... other files ...
 *
 * Berkeley-based Unix
 *              % cc -DBSD program.c ... other files ...
 *
 * PC and Microsoft C [timing granularity 1/18 second]
 *              C:\> cl -DPC program.c ... other files ...
 *
 * PC and Microsoft C [timing granularity 1/1000 second]
 *              C:\> cl -DPC -DFAST_CLOCK program.c ... other files ...
 *
 */

#define DEBUG                                   /* Turn on timing statistics */

#define TIX_DIFF                        (_t2 - _t1)

/***************** For Berkeley Versions of Unix ****************/

#ifdef BSD
#       include <sys/types.h>
#       include <sys/time.h>
        static double _t1, _t2;
        static struct timeval _tv_;
        static struct timeval _timez;
#       define gettime()        gettimeofday(&_tv_, &_timez_)
#       define START()          {gettime(), _t1 = _tv_.tv_usec/1e6 + _tv_.tv_sec)
#       define STOP()           {gettime(), _t2 = _tv_.tv_usec/1e6 + _tv_.tv_sec)
#       define PRINT_TIME(cp)   fprintf(stderr,"%s: %lf secs.\n",cp,TIX_DIFF)
#endif

/***************** For System V Versions of Unix ****************/

#ifdef SYSV
#       include <sys/types.h>
#       include <sys/times.h>
        static struct tms _currtime;
        long times();
        static long _t1, _t2;
#       define gettime()                times(&_currtime)
#       define TIX_PER_SEC              60.0            /* Verify for Your Machine */
        /* For the 3B2 use 100.0 */
#       define START()                  _t1 = gettime()
#       define STOP()                   _t2 = gettime()
#       define PRINT_TIME(cp)   fprintf(stderr,"%s: %lf secs.\n", \
                                                cp, TIX_DIFF / TIX_PER_SEC)
#endif

/***************** For the PC ****************/

#ifdef PC
#       ifdef FAST_CLOCK
#               define TIX_PER_SEC                      1000.0
#       else
#               define TIX_PER_SEC                      18.20648
#       endif
        unsigned long gettime();
        static long _t1, _t2;           /* Tick values for calc. elapsed time */
#       define START()                  _t1 = gettime()
#       define STOP()                   _t2 = gettime()
#       define PRINT_TIME(cp)   fprintf(stderr,"%s: %lf secs.\n", \
                                                cp, TIX_DIFF / TIX_PER_SEC)
#endif

#ifdef DEBUG
#       define TIME(code, string)       START(); code; STOP(); PRINT_TIME(string)
#else
#       define TIME(code, string)       code
#endif
```

Listing 2

```
 1  /*
 2   * clock.h --  (real) Quick and (really) Dirty macros
 3   */
 4
 5  #define MSN(x)                       ( (x) >> 4 )
 6  #define LSN(x)                       ( (x) & 0x0f )
 7  #define LSB(x)                       ( (x) & 0x00ff )
 8  #define MSB(x)                       ( (x) >> 8 )
 9
10  #define BCD(x)                       ( (((x) / 10) << 4)  |  ((x) % 10) )
11  #define UNBCD(x)            (MSN(x)*10 + LSN(x))
12
13  #define CLOCK_RATE          1193180          /* 1,193,180 Hz or 1.19 MHz */
14  #define STD_PULSE           -1                       /* 65536 or 0xFFFF */
```


V-LIB

the user interface library for C

V-LIB is a comprehensive, easy to use library of over 150 custom C functions for building sophisticated PC applications.

Windows • *overlapping, tiled, built-in window management*

Menus • *vertical, horizontal pulldown, popup, arrays*

Forms • *full screen or popup data entry with multiple field types*

Pop Ups • *messages, prompts, selection lists, scroll bars*

Formatted Screen Output

Full Editing Input

High Speed Display

... and much, much more!

Give your programs a professional look! Develop applications faster!

Compilers supported: Microsoft C and Quick C, Borland Turbo C, Lattice C, and Datalight C.

All memory models supported.

Demo disk with usable sample programs, source, and reference card . . . **$10**

Library with reference card and 280 page programmer's manual **$99**

Library (as above) with full source code **$149**

Prices include shipping. California residents please add 7% sales tax.

No royalties. Site license available.

Call or write:

Pathfinder Associates
291 Madrone Avenue
Santa Clara, CA 95051
(408) 984-2256

Visa and MasterCard accepted.

BBS: (408) 246-0164 (1200/2400 N-8-1)

Number 145 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Eco-C88 C Compiler with Cmore Debugger

Professionals prefer the Eco-C88 C compiler for ease of use and its powerful debugging features. Our "picky flag" gives you nine levels of lint-like error checking and makes debugging easy.

"I'm very impressed with the compiler, editor, and debugger. I've tried quite a few different compilers for the PC and have given up on all of the others in favor of yours... I've gotten to the point where I download C code from a DEC VAX/VMS system just to be able to compile it with the picky flag set at 9. It finds lots of things VMS totally ignores..."
JS, Oak Ridge, TN

The Eco-C88 compiler includes:

- A full-featured C compiler with 4 memory models (up to 1 meg of code and data) plus most ANSI enhancements.
- Without a doubt, the best error checking you can get. We catch bugs the others miss, making you much more productive.
- Cmore is a full-featured source code debugger, not some stripped-down version.
- Robust standard library with over 230 useful (no "fluff") functions, many of which are System V and ANSI compatible. Full source is available for only $25.00 at time of order.
- CED, a fast, full screen, multiple-window program editor with on-line function help. You can compile, edit, and link from within CED.
- cc and mini-make utilities included that simplifies the most complex compiles.
- Users manual with over 150 program examples (not fragments) to illustrate how to use the library functions.
- Fast compiles producing fast code.

Our Guarantee: Try the Eco-C88 compiler for $99.95. Use it for 30 days and if you are not completely satisfied, simply return it for a full refund. We are confident that once you've tried Eco-C88, you'll never use anything else. Call or write today!

Orders: **1-800-952-0472**
Info: **1-317-255-6476**

Ecosoft Inc.
6413 N. College Avenue
Indianapolis, IN 46220

ECOSOFT

Number 8 on the Reader Service Card

Greco Code, Continued


```
15  #define RESTORE_PULSE()     setpulse(STD_PULSE);
```

```
 1  /*
 2   * test.c -- Driver program to show how to use new timing facilities
 3   *
 4   * For the PC, compile as follows:
 5   *
 6   *         C:\> cl -DPC -DFAST_CLOCK test.c gettime.c setpulse.c fixtix.c
 7   */
 8
 9  #include <stdio.h>
10  #include <process.h>
11  #include <dos.h>
12  #include "timer.h"
13  #include "clock.h"
14
15  #define NO_ERR              0               /* No error to return to the shell */
16  #define NO_ARGS     1                   /* If nothing to time, return this value */
17
18  main(argc,argv)
19  char **argv;
20  {
21     char *bp;                           /* Ptr to command to be spawned */
22     unsigned long gettime();           /* Gets clock ticks from MS-DOS */
23     int i;
24
25     if (argc == 1)              /* If nothing to time, bye-bye */
26             exit(NO_ARGS);
27
28     bp = argv[1];
29
30  #ifdef PC
31     setpulse(1000); /* Set timing granularity to 1/1000 of a second */
32  #endif
33
34  /* Test #1 */
35     TIME(spawnvp(P_WAIT, bp, ++argv), bp);
36
37  /* Test #2 */
38     START();
39     for (i=0; i<250;i++)
40     {
41             printf("\r  %d  ", i);
42     }
43     STOP();
44     PRINT_TIME("for loop");
45
46  /* Test #3 */
47     TIME(printf("Hello World\n"), "printf call");
48
49  /* Test #4 */
50     TIME(func1(), "Call to func1() using integer variable");
51
52  /* Test #5 */
53     TIME(func2(), "Call to func2() using register variable");
54
55  #ifdef PC
56     setpulse(STD_PULSE);
       /* Reset timing granularity back to 1/18 of a second */
57     fixtix();                           /* Adjust BIOS view of time */
58  #endif
59     exit(NO_ERR);               /* Good Housekeeping */
60  }
61  /*****************************************/
62  func1()
63  {
64     int i;
65
66     for (i = 0; i <1000; i++)
67             ;
68  }
69  /*****************************************/
70  func2()
71  {
72     register int i;
73
74     for (i = 0; i <1000; i++)
75             ;
76  }
```

```
 1  /*
 2   * gettime.c  --   Uses BIOS interrupt 0x1A to get the timer count
 3   *                                 in clock ticks.
 4   */
 5
 6  #include <dos.h>
 7
 8  #define             TOD_INT         0x1a
 9
10  unsigned long gettime()
11  {
12     union REGS in, out;
13     unsigned long current_tix;      /* 32-bit tick value to be returned */
14
15     in.h.ah = 0;                                /* AH = 0    GET clock ticks */
16     int86(TOD_INT, &in, &out);      /* Call BIOS timer tick interrupt */
17
18     current_tix = ( (long) out.x.cx << 16 ) + out.x.dx;
19
20/* If midnite passes, add a day's worth of clock tix to the return value */
21
```

Code Continues...

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programmer's Tools

from The Software Family

Data Base

Btree/ISAM *by Softfocus*: The btree index structure provides rapid access to data based on content with multiple keys per file. Includes full C source code. ISAM automatically keeps data and indices synchronized, reducing your work compared to other btree packages. Multi-user functions available. For use with all popular C compilers. $99

QBase *by Crescent Software*: Qbase is a screen designer and relational database for BASIC that lets you quickly create applications including data entry, validation and file management. Qbase data validation is controlled by the database, so all input edits are automatically changed when you modify the database definition. Includes all source code. Specify QuickBasic or TurboBasic when ordering. $79

386 Memory Management

386^{MAX} by Qualitas: Exploits the 80386's memory management capabilities to free DOS memory for applications and maximize use of the fastest system memory. Makes up to 640K available for programs. $69

Windows and Data Entry

Vitamin C *by Creative Programming* provides a comprehensive set of C functions for windows, menus, data entry & validation. Includes complete C source code. Specify your primary compiler when ordering. $159

Artificial Intelligence

Smalltalk/V *by Digitalk*: Smalltalk is pure object oriented programming. By classifying objects and defining interactions, you model systems in terms matching your own thinking and language. The advanced graphical interface duplicates the fast, responsive feel of workstations costing far more. $82

New! Smalltalk/V286 w/multi-tasking. $164

Debugging and Quality Control

Periscope II *by Periscope*: This is the debugger that lets you know where your program is getting hung up, even if you don't know where to set the breakpoint! Just press the NMI hardware button provided with the debugger to interrupt the current program at any time and enter the debugger. Don't waste hours guessing where to place you breakpoints or make do with less complete debuggers. For all popular languages and compilers. $139

PC Lint *by Gimpel*: Identifies hard-to-find C errors like inconsistent declarations in multi-module programs and "=" instead of "==" in expressions. For all compilers. $95

Other Products

Microsoft C v5.1	$279
Microsoft Macro Assembler	$98
Microsoft QuickBASIC v6.0	$68
Borland Turbo C v1.5	$68
Borland Turbo Pascal v4	$68
Blaise C Asynch Manager	$125
Blaise C Tools Plus/5.0	$95
Blaise Turbo Power Tools/4.0	$95
Essential C Graphics	$195
Essential C Utility Library	$130
Essential Resident C	$169
Essential Screen Star	$169
Pharlap ASM/Link	$389
Santa Cruz Operations Xenix/V	$479
Digitalk Smalltalk/V EGA/VGA	$43
Digitalk Smalltalk/V Comm	$43
Digitalk Goodies #1, #2, #3 each	$43

Free Catalog & Newsletter

The Software Family carries hundreds of products from dozens of publishers. Call or write for your free copy of our comprehensive catalog and newsletter.

Returns

The Software Family wants you to pay only for the tools you really use. If you order anything featured in this ad (or most of the products in our catalog) you can open and use the product for up to 30 days and still return it if it doesn't meet your needs.

The Software Family

800-443-7176

(415)957-0111

Visa • MasterCard • COD
AmEx • Discover • Check

- Most orders shipped within 24 hours
- Free UPS shipping on orders over $100 ($3 on orders under $100)
- Federal Express, UPS air, and international service charges: our cost + $3 handling
- $2 COD fee
- No surcharge for credit cards

The Software Family
649 Mission Street, Dept PJ058
San Francisco, CA 94105

Technical Support

The Software Family carefully evaluates features, quality and support before offering a product to our customers. All of the products featured in this ad are backed by excellent technical support from the publisher — in many cases directly from the program developers. All are published by companies with reasonable update policies and with a track record for responding to customer requests for both bug fixes and new features. We at The Software Family further back the products with our own technical support. We are willing and able to intercede with the publisher if your needs aren't satisfied or to research problems caused by the interaction of products we recommend. We can also provide quotes for custom modification of source-licensed products, in-depth technical consultation or custom-developed products.

Prices subject to change without notice - ©1988 TSF All Rights Reserved

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

#1 C Interpreter

C-terp

Develop C programs with the safety, and convenience that only our highly acclaimed C interpreter can provide.

"C-terp is an excellent way to develop C programs. It has a good user interface and never went astray.

Ken Pugh, The C Users' Group

"C-terp comes closest to this ideal [C development environment]"

John Unger, BYTE

"... easy to use, powerful, and a time saver ... one of my favorite production tools"

Larry Jordan, The C Journal

Not just a learning tool: Large memory model, multiple-modules, a multi-file, reconfigurable editor, automatic make, virtual memory option, shared symbols option, complete object code compatibility, links in your compiler's entire library.

Full screen debugging: single-step; side-step; display variables, arrays, structures; watch expressions; watch conditions; execute any C expression (even with macros); traceback; leave-function; sticky, temporary and cursor-directed breakpoints. Out-of-bounds pointers are trapped.

Even Bill Gates says there's nothing like an interpreter.

Gimpel Software

3207 Hogarth Lane
Collegeville PA 19426
(215)584-4261

For MS-DOS, specify compiler
$298.00 Microsoft 4/5, Lattice, Aztec, C86, C86 Plus, Mk Wms
$139.00 Turbo C 1.0/1.5
$498.00 Xenix

Order Today - MC, VISA, COD

PA residents add 6% sales tax. Outside USA add $15. C-terp is a trademark of Gimpel Software.

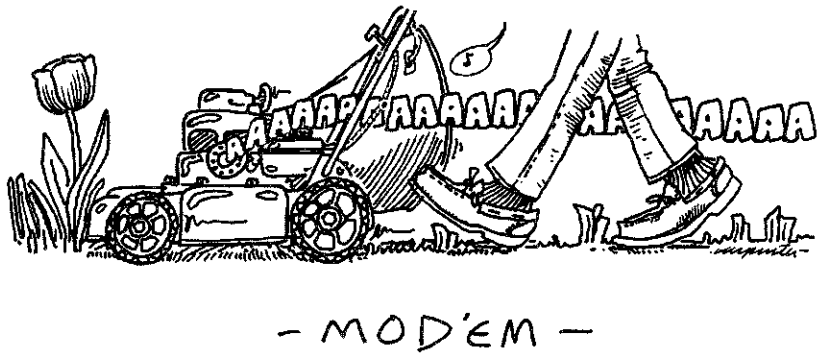
Contact this advertiser directly


```
22      return ( out.x.ax & 0xff ) ? current_tix += 0x01800b0L: current_tix;
23  }
```

```
 1  /*
 2   * fixtix.c  --  Reset the BIOS view of the clock ticks.
 3   *               Needed after programming the timer chip to pulse faster.
 4   */
 5
 6  #include <dos.h>
 7  #include <stdio.h>
 8  #include "clock.h"
 9
10  #define AH          regs.h.ah
11  #define CH          regs.h.ch
12  #define CL          regs.h.cl
13  #define DH          regs.h.dh
14  #define DL          regs.h.dl
15
16  union REGS regs;
17
18  fixtix()
19  {
20      long far *fp;
21      long hrs, mins, secs, hsecs, total, tix;
22
23      FP_SEG(fp) = 0x0040;
        /* Point to BIOS data area where tix are stored */
24      FP_OFF(fp) = 0x006C;
25
26      AH = 2;                         /* GET real time clock */
27
28      int86(0x1a, &regs, &regs);
29
30  /* Convert everything to 1/100 seconds */
31
32      hrs = (long) 60 * 60 * 100 * UNBCD(CH);
33      mins = (long) 60 * 100 * UNBCD(CL);
34      secs = (long) 100 * UNBCD(DH);
35      hsecs = (long) UNBCD(DL);
36
37  /* Total them up and convert to clock ticks */
38
39      total = hrs + mins + secs + hsecs;
40      tix = (long) total * .1820648;
41
42  /* "Poke" the tick value into the BIOS data area */
43
44      *fp = tix;
45  }
```

```
 1  /*
 2   * setpulse.c  --  Will program the 8254 to Pulse at a desired rate.
 3   */
 4  #include "clock.h"
 5
 6  setpulse(num)
 7  int num;                            /* desired timing resolution */
 8  {
 9      int ax;
10
11      ax = (num == STD_PULSE ? 0xFFFF : (CLOCK_RATE / num));
12
13      outp( 0x43, 0x36);              /* 0011 0110 */
14      outp( 0x40, LSB(ax) );  /* First the Least Significant Byte */
15      outp( 0x40, MSB(ax) );  /* Then the MSB */
16  }
```

◆



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Works with MS & TurboC

If you need the best source debuggers for MS-DOS or embedded microprocessor development with excellent support for C, PLM86, and assembly language contact:

SoftAdvances

10811 Washington Bl., Ste. 205 • Culver City, CA 90232 • (213) 559-7015

Number 33 on the Reader Service Card

Attention Programmers:
Affordable display utilities.

Now you can make screens, demos and graphics quickly and easily. And improve your competitive position at a price you can afford.
Satisfaction guaranteed or your money back!

"Tremendous productivity tools"
*(**PC Week** October 27, 1987)*

"Many applications will benefit from the quick development times and minimal programming required to implement a finished, quality product"
*(**IEEE Software** November, 1987)*

"If you want to include bit-mapped graphics in your application, then this might be the package for you"
*(**Computer Language** June, 1987)*

DisplayExpress™ $69
A fast, efficient and easy-to-use screen manager.

C-Display Librarian™ $69
Professional video, keyboard and graphics functions.

Send check or money order now to:

SYDETECH
SYSTEM DEVELOPMENT TECHNOLOGIES, INC.
43-23 COLDEN STREET, #17C, FLUSHING, N.Y. 11355

For COD orders or the name of the nearest dealer please call (718) 886-0221

Number 74 on the Reader Service Card

Unbelievable!

SOURCER™

Creates commented source code and listings from memory, .COM files or .EXE files

- **CLARIFY UNDOCUMENTED CODE**
- **EASILY MODIFY PROGRAMS**
- **RECREATE LOST SOURCE CODE**

SOURCER™ creates detailed commented listings and source code directly suitable for assembly. Built in data analyzer and simulator resolves multiple data segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines all necessary assembler directives. Complete support for 8088 through 80286, V20/V30, 8087 and 80287 instruction sets. No other product comes close to the output quality of SOURCER.

BIOS SOURCE

PS/2 ▪ AT ▪ XT ▪ PC ▪ Clones

- **CHANGE & ADD FEATURES**
- **CLARIFIES BIOS INTERFACES**
- **SPECIFIC TO YOUR MACHINE**

The bios pre-processor to SOURCER™ provides the first means to obtain accurate legal source listings for any bios! Identifies entry points with full explanations. Resolves PS/2's multiple jumps for improved clarity. Provides highly descriptive data labels such as "video_mode" and "keybd_q_head," and much more. Fully automatic.

SOURCER™	$ 99.95*
SOURCER™ with BIOS Pre-Processor	$139.95*

(*OUTSIDE USA, ADD $15 SHIPPING; CA RESIDENTS, ADD SALES TAX)

All our products come with a 30 day money back satisfaction guarantee. Not copy protected. To order or receive additional information just call!

800-538-8157 x 811 (outside Calif.)
800-672-3470 x 811 (inside Calif.)

V COMMUNICATIONS
3031 Tisch Way, Suite 200, Dept. J1
San Jose, CA 95128
(408) 296-4224

PS/2, AT, XT and PC are trademarks of IBM Corp.

Number 125 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Getting Your Consulting Tools in Order

by
Paul Barkley

I've been busy lately trying to upgrade my consulting tools. Six years ago I was far better automated than my clients. Hardware and software have been improving steadily during the last six years, of course, and while the software that I create for clients is far more sophisticated today, I haven't done much with my own tools. I'm getting further behind every day.

I decided it was time to spend some effort (and some money) on searching for and acquiring new tools. Business had been very good lately (knock on silicon), and I thought a few tax write-offs wouldn't hurt. More importantly, I'd finally reached the point where I was discontented; I didn't want to continue plodding along with my old tools.

Teamwork/PCSA

I started by taking a look at an interesting piece of software that helps you design structured systems. I firmly believe that the days of hand coding are drawing to a close; it's time to get on with the process of automating design and coding. For some time, I've been watching ads that show lovely computer drawn data flow diagrams (DFDs), and I decided to check out a few of them. Cadre Technologies, Inc., was kind enough to let me review their product, Teamwork/PCSA.

First, my bias. I don't know beans about the sort of formal structured analysis that produces data flow diagrams. My review of Teamwork/PCSA started with the question, "Can a computer consultant with no background in data flow diagrams use an automated package to enhance his or her consulting practice?" My emphasis was on whether the package could provide any practical benefits.

I made a mistake. I started with the Teamwork tutorial. The tutorial takes you through part of a design for a speed control in an automobile. When I told my contact at Cadre Technologies that I ran through the tutorial but didn't have the slightest idea what I'd "designed," she asked me whether I had speed control in my car. I may have said something rude. I knew exactly how speed control works in a car; I just didn't understand data flow diagrams.

If you don't know how data flow diagrams work, and why they're extremely useful, you not only won't be able to do any useful work with the package, but you won't understand even the examples.

Now the good news. Teamwork is packaged with a copy of Tom DeMarco's book, *Structured Analysis and System Specification* (Yourdon Press, 1978). If you don't know what data flow diagrams are all about, start with the book. It's a lively read and presents the concepts clearly. The necessity for each element of structured analysis and design is clearly rooted in the practical world; DeMarco's experience is evident in the problems he discusses. If you have a practical mind and like practical solutions, the DeMarco book makes a compelling argument for the methodology.

We don't have time to discuss the merits of structured analysis. I'm not a rabid follower, but it's hard to argue against the system when it makes so much sense. Let's just say that it provides a method of dealing with large systems in an iterative fashion. It traces the flow of data rather than the flow of control; this idea takes some getting used to. Systems are shown as levels starting with the top, bird's-eye view; diagrams descend to increasingly more detailed levels. The breaking down activity stops when processes that cannot reasonably be further partitioned are identified.

The power of DFDs is that they are meant to be used in an iterative fashion. The analyst draws a diagram, hooks pieces together, and then discusses it with the users. Problems are discovered, and the diagrams are redrawn. Eventually, a system emerges.

Teamwork automates the process of drawing nice diagrams. Circles and other symbols are drawn properly without any effort on the part of the analyst. Lines are easily moved, and tags attached.

Teamwork, however, goes far beyond this to check the accuracy of the design. As you read you'll discover that DeMarco takes you down the primrose path. You wind up nodding "yes, yes" as you follow his designs until he stops and says, "Now what's wrong with what we just put together?" If you're catching on to the process, you'll be able to spot the errors. DeMarco then shows you how to correct those problems.

One of the best features of Teamwork is that it will perform a great deal of error checking for you. For example, since data flows ultimately must be described at the data dictionary level, it's important to make sure that data required in a process be available to the process. In other words, if it flows out, it has to have flowed in from somewhere. It's easy to miss something small like that.

At a cruder level, it's important if the diagram shows a certain number of flows that the same flows be present when you drop to a more detailed level. If not, there's a mistake.

My experience with analysis (and maybe with everything else in the world for that matter) is that you take two steps forward and one step back. The only way you ever get anything analyzed or designed is to take a stab at it; you keep working on it, making it better, until one day you're finished. "Tuna doesn't just swim into the can."

Initially, I wasn't very impressed with Teamwork's ability to draw shapes. "Big deal," I thought. I've got a flow charting package that can handle a lot more shapes than Teamwork, and it seems a lot easier to use. I can also grab a template and manually draw data flow diagrams pretty easily; in fact it's kind of fun for a change to do something with a pencil instead of a keyboard. So, drawing wasn't big on my list of great features.

Enter the Mouse

Nor was I impressed with the mouse interface. I have to confess, I'm not a mouse user. I never could figure out why anyone would want to take his or her hands off the keyboard to fool around with a device that leaves a messy cable strung across the desk.

Unfortunately, Teamwork requires a mouse, so in order to look at the package I had to go buy one of the critters. Bah. My first reaction was that a clever program would

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

be able to make use of a mouse but not make a mouse mandatory. My esteem for Teamwork dropped a little.

One problem with mice, I found, is that you're expected to know where to move the mouse cursor and when to click the button. Because I can't even remember what day it is, I like menus that take users through an application; a well designed menu system can be very fast. Trying to remember that clicking the button while I'm pointing to an obscure little box in the corner of the screen (totally unlabeled of course because it's a very small box)—well, trying to remember this sort of stuff was distracting. The worst part was using the mouse to work on the data dictionary. You have to position the mouse cursor inside the carat cursor; this requires a steady hand and a lot of false clicks.

But since I had the mouse, I tried it with a few other packages.

I couldn't figure out why there was a big ugly box character in the middle of my screen when I wrote a short little program in Microsoft's QuickBASIC. Every time I started QuickBASIC, this odd box would appear in the middle of my screen. Another consultant, looking on while we patched together a quick program, managed to bump my mouse and send that odd box flying. "Ohhhh . . . " we said, light bulbs coming on upstairs. "Now we understand. The box is the mouse cursor. Neat!"

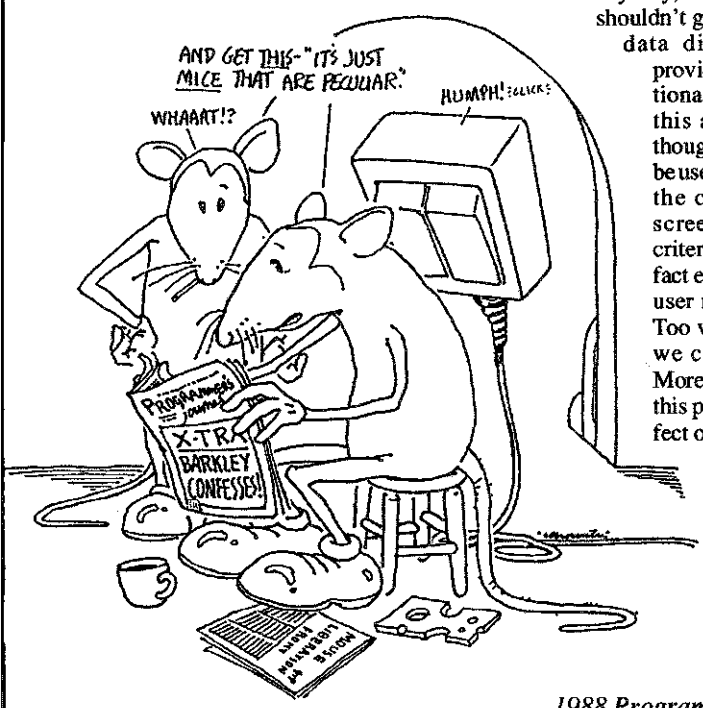
I'd forgotten that Microsoft had a mouse very much in mind when they put QuickBASIC together. Practicing a little with the mouse removed some of my reservations about Teamwork's user interface. Once you get a little practice with a particular program and the mouse, it's not difficult. Still, I don't like interfaces that require me to remember anything at all; I like to have clear prompts always available. I decided that Teamwork's use of a mouse wasn't so weird after all; it's just mice that are peculiar.

Enough about mice. Back to structured data flow diagrams.

DeMarco makes a point about data dictionaries that illustrates one of my few reservations about Teamwork. DeMarco says that although the data dictionary can be expected to survive the analysis phase, he doesn't want to put too much emphasis on that. Some analysts put all sorts of detail into the dictionary, and DeMarco thinks this can easily get in the way of the analysis and design phase.

My attitude—part of my original premise for looking at the product—is that it should be possible to build a coherent set of tools that takes the automation process from analysis and design through coding, documentation, and maintenance. This, however, is a pet subject of mine, my holy grail. The creation of an automated software factory, I realize, is pretty ambitious stuff.

Anyway, DeMarco says that you shouldn't get carried away with the data dictionary. Teamwork provides an adequate data dictionary facility that matches this attitude. What I want, though, is a dictionary that can be used to create files and drive the creation of data entry screens by specifying edit criteria automatically, and in fact even drive the creation of user manuals. Yeah, I know. Too wild. Let's stick to what we can accomplish today. More importantly, let's not let this pie-in-the-sky attitude affect our evaluation of the usefulness of Teamwork.



Teamwork works smoothly and has lots of nice features that keep you from making design errors. The only other bad thing I can say about it is that it comes with a hardware key that must be plugged into your serial port. The hardware key is designed to allow a serial device to continue to use the port to which the key is attached; it simply passes data through to the printer or modem as needed. Apparently, at start up Teamwork checks to see if the key device is present. There aren't that many programs left that have copy protection, so all that effort seems wasted. The hard-ware key worked fine although I didn't test it with anything attached to the same port.

So, have I damned Teamwork with faint praise? No—I think Teamwork does an excellent job of producing data flow diagrams. If you're comfortable with the concept of DFDs (and you like mice), you'll find that Teamwork will beat the pants off drawing and re-drawing DFDs by hand. Simply put, Teamwork will greatly increase your productivity. I found no bugs, and Teamwork has lots of nice features to prevent making mistakes. At $995 you should be able to recover your costs on the first few jobs for which you use it . . . and just think what users will say when they see all those great diagrams!

Mouse Resistant

I've never been able to figure out why it takes me so long to catch on to new things. Seems like everyone else is using the latest technology while I'm still puttering along trying to get my work done with yesterday's tools.

I've mentioned several times that one problem with using new tools is that you must take the time to install them and practice with them before you can start getting useful work out of them. We all want the latest and greatest, but sometimes when we get new goodies large amounts of time disappear into some temporal black hole.

I mentioned that I had to buy a mouse in order to try out Teamwork. It took me a few days (well, off and on) to figure out whether I wanted to buy a serial mouse or a bus mouse. Hmmm. Did I want to use up a serial port, or did I want to use up a slot? I decided I could spare a slot.

Now, did I want a two button mouse or a three button mouse? Jeez, how should I know how many buttons I need on a mouse? I've never used one. It looked to me as if there were more two button mice around. Did I want a mechanical mouse, an optical mouse, or a mechano-optical mouse? Unbelievable! All I wanted was a device that would let me try out this software package, and maybe use it later for desktop publishing.

Continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

So what did I do? Oh, I did my homework—pulled out the back issues of magazines and read up on mice. Then I went shopping for the mouse rated the best. No one had it, but everyone recommended another major (as in *very* major) brand, and in the end I said the hell with it and bought the one everyone recommended (and which they just happened to have in stock). I didn't figure a mouse should be the subject of four months of my time.

Then I spent a couple hours unpacking the mouse, reading the manual, and messing around with the jumpers. When I installed the thing it killed my machine, and it took a couple tries before I got the jumpers right. Hey, that's a hardware problem; I just do software. How should I know what interrupts the other stuff in my machine is using? Of course the solution was easy; it could have been installed by your average nine year old in about ten minutes, assuming the nine year old had installed one before.

Next I had to try out the demos, including the graphics package that was included with the mouse. That shot another couple hours while I proved to myself that although I now have a whole pile of neat stuff like paint brushes and spray cans and erasers, just like a Macintosh, I don't have any more drawing ability than I had when I was in second grade. Well, it was fun anyway.

So, by the time I finished doing research, running around to buy the mouse and installing it, I probably spent the equivalent of a whole work day. Expensive little critter; glad I don't have to feed it or change its litter.

I spent the equivalent of another full day explaining to my colleagues why I paid $20 more than they did for a mouse obviously inferior to the one they bought, explaining what in the world I was thinking of when I bought a mouse with the wrong number of buttons, and trying out tons of mouse menu software that they sent me and insisted that I couldn't live without. Remember the old stomach remedy ad that said, "Try it, you'll like it?" I tried it. I didn't like it. The mouse menus made just about every piece of software that I own run much worse. They *were* kind of cute, though.

Okay, so a mouse is just a little tool. I have it now, and when I feel the need to create some Grandma Moses graphics of little houses or sailboats that I will be unable to print with any hardware or software that I own, I'll be all set.

This is all leading up to something much worse. The mouse was just the first salvo.

Shopping for a 386

I firmly believe that every consultant must be prepared to use the most efficient means possible to produce reports, systems, and (most importantly, of course) client bills and thus must be automated to the hilt. I've been limping along with an original IBM PC with the 64K motherboard. I've upgraded it many times with memory, printers, and hard disks. When the ATs came out, they seemed too expensive to me for what they did, so I waited. By the time they got reasonable (to my mind), the 386s were already out. So I waited a while, figuring I might as well make a big disruptive jump as a small disruptive jump.

I did my research, as usual, and when I started shopping for 386 machines I looked for a few second-tier machines; you know, machines from small, but recognizable, clone makers. I haven't been able to figure out why anyone would want one of the new IBM PS/2 machines—except perhaps to display an electronic simulation of a Rolex watch with Cheryl Tiegs superimposed. (I saw this stunning graphic running as a demo, no doubt unauthorized by either the lady or the watchmaker.)

Anyway, you know how backward I am. I'll wait until everyone has PS/2s (assuming that happens one of these days), and then I'll join the mainstream. In the meantime, what I need is a machine that simply extends the kind of hardware platform I've been using. (I bet I get some nasty letters about this from PS/2 fans.)

While I was out shopping for 386 (16 megahertz) machines, the 20 megahertz models started to hit the streets, and by the time I completed my purchase, I got the faster processor for about the same price. Part of that decision involved accepting a totally generic 386 machine, something that seems a lot safer today than accepting a generic 8086 machine a few years ago. The case came with a little slot obviously designed to grab a clip-on name tag. Since my machine was so generic the dealer didn't even want to put his name on it, I now have a "Barkley 386." That's a little too cute, I know, but what are you going to do with the hole in the case otherwise?

This is all leading up to something much worse. The mouse was just the first salvo.

I won't go into all the gory details, but of course I got a new multi-syncing monitor and a snazzy new high resolution EGA card while I was at it, not to mention a larger disk drive with an RLL controller. All my choices seemed fairly cost-effective to me. My dealer had to take the new machine back to *his* dealer five times to get the disk drives working; after they replaced the hard disk once, and the controller card once, and did absolutely nothing at all a couple times, they discovered a bad cable between the controller and the hard disk. Then they got around to installing the extra 360K drive, and the system died again; they couldn't find the right jumper to enable it. Needless to say, my dealer was messing with their first 20 megahertz machine and

Form-Fill Toolkit

Product: C Library
Price: $59.95
Features:

Cursor Control
Sound & Printer Control
Fast Screen Access
Keyboard Access
Edited Field Entry
Pop-Up Window Support
Display Text Windows
Menu & List Selections
Form-fill Routines

Compiler Support
Microsoft 4.0 : Y
Microsoft 5.0 : Y
Quick C : Y
Turbo C : Y
Other:

↑↓ Scroll F1 Help F2 Save F4 Cancel Esc No Save

Provide a consistent user interface for your programs with over 100 documented functions.
SPECIAL! Includes powerful multi-record data collection module with fully commented pristine source code.

Anji Software, Inc.
P.O. Box 297
Cedar Grove, NJ 07009
(201) 239-4328

• Library Source Code - $119.95
• Specify compiler when ordering
• No Royalties required
• Demo Diskette $10.00

We accept Visa and M/C
NJ Residents add Sales Tax
Shipping $3.00 Domestic
$10.00 Foreign

Number 162 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

with their first installation of that particular drive and controller.

Configuring the 386

Once the dealer got his act together (and boy was I glad that *he* was responsible for the system and not I), I had no difficulty with the disk drives. All I had to do was load the disk up again with all the stuff from the old system; I probably spent three days digging stuff out of the file cabinet. It seemed like a good time to rearrange some of my software and load all the great stuff I've been keeping offline. Funny how fast all that storage filled up.

Beyond this, there was the time I spent trying to talk WordPerfect into working with the marvelous high resolution graphics modes now available to me. I finally gave up and called the board maker. They cheerfully told me how to get everything working; once again the solution was easy once you learned the trick.

I didn't realize that WordPerfect was smart enough to let the user specify the number of rows and columns on the screen. I was impressed since I expected to have to make some awful DEBUG patch to get it to work as in the good old WordStar days.

The software supplied with the board is menu driven; there are about 60 different modes that can be selected. Holy overload, Batman! It can take you fifteen minutes just to study the display selections. So, selecting a mode is simply a matter of picking it off the display. DOS works just fine in any mode you select.

I was interested in changing the default configuration because my printer (that's another sore subject—we'll get to that in a minute) packs so many characters on a line of proportional type that I have to set margins wider than 80 characters to make output look right on paper. Obviously, I can't see a full 90 character line on the screen without horizontal scrolling. Also, I thought it would be real nice to be able to see a full page instead of only 24 lines.

After I browsed through the choices, rejecting those with resolutions *lower* than the EGA's, it became apparent that I could set up the display for one of the following:

80 x 25
80 x 60
80 x 66
94 x 29
100 x 75
100 x 60
132 x 25
132 x 29
132 x 32
132 x 44

I tried out a bunch of them just for fun, and they worked great. Each one was very readable on my monitor; well, each was very clear, but it sure was teensy. I'm blessed with great eyesight, and I'd like to keep it, so I ruled out most of the sizes. I finally settled on 94 x 29 because it's just wide enough to show a single line in the margins I require. I still find it greatly amusing to fire up my "special" version of WordPerfect and have it appear in that size. It's handy, too.

Again, all this configuration stuff is a big pain. Sure, it's entertaining. Most of us enjoy messing around with new stuff, but when clients are calling and want to know when you're going to send their software, it takes away some of the fun. I was a little embarrassed a few days ago when I had to tell a client I'd better mail their application update because my communications software was no longer working. Not only did seconds fly by on the data comm display about ten times too fast, but nothing else seemed to work at that speed either. Ralph Nader might have said that that particular communications package was unsafe at any speed, but it sure won't run right on a 386.

I forgot to mention that I got rid of the old system a few days before the new one arrived, thanks to a scheduling snafu. I had already committed the old system. Seems to me this situation is exactly what we try to avoid when we run parallel systems for clients. Naturally, we consultants are so smart we don't have to do things for ourselves the way we do things for clients; for example, who designs his or her own systems before coding?

The Payoff

Now, having complained about all the time lost to setting up the new system, I have to point out one advantage. It used to take me about 20 minutes to compile a program and link it with the other object code in a large Clipper application. I tried the same thing

The DBMS that Rewards Expertise

You are an experienced application developer. A specialist. You have worked hard to get here. And you are looking for a DBMS that will let you exploit your expertise. Not waste it. You know that all the popular DBMSs out there are designed for the masses — anyone can use them. You know that there are no miracles: a system that is best for you, could not possibly be used by novices.

Meet ***The Andsor Collection***: the only DBMS that does not waste your talents. Many find it difficult. Because it was designed for experts. It uses a unique concept, a combination of built-in operations and procedural language, an environment where the application logic and data are kept together and are handled in the same way. It baffles novices, but is incredibly powerful in the hands of experts. If you can tame its power, a rich reward awaits you: *fully customized applications with one tenth the effort*. You deserve it. Because you have worked hard to get here.

- A self-contained, interactive environment, where you can build and modify an application even while it is running.
- Programs *one tenth* their size in other systems.
- Windows, screens, help, menus, with practically no code at all.
- A new, open-ended way of relating files, can simplify applications by creating relations that are impossible in other systems.
- Variable code that changes at run time, lets you perform more than one operation in the same program section.

If you are an average programmer, use the popular DBMSs: they have sold millions, and you need this comfort. But if you think you are a tough programmer, accept this challenge. Write for our brochure and ***FREE*** demo disk. It contains two sample applications, also showing design and programming details. There are comments and some help, but no reference manual. Study the applications. Then, if you can understand more than half, congratulate yourself: you are ready to step up to the one DBMS that rewards expertise.

The Andsor Collection™

NOW IN ITS THIRD YEAR

$145 *after June 30, 1988* **$295**

60 DAY MONEY BACK GUARANTEE

Visa, MC, AmEx, Check

ANDSOR®
ANDSOR RESEARCH INC.

390 Bay Street, Suite 2000
Toronto, Ontario M5H 2Y2
(416) 245-8073

To order ***The Andsor Collection*** call toll free (U.S. and Canada)
1-800-628-2828 Ext. 535

Price includes shipping in the U.S. and Canada. Please add $10 for shipping to other countries. If you return the software, $8 will be deducted from the refund, to cover our shipping cost.

System requirements: any IBM PC or PS/2 or fully compatible, 320K RAM, one drive or hard disk, DOS 2.0+ or OS/2.

© 1988 Andsor Research Inc. Andsor is a registered trademark and The Andsor Collection is a trademark of Andsor Research Inc.
IBM is a registered trademark and IBM PC, PS/2, OS/2 are trademarks of IBM Corporation.

Number 191 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

with the new system, and it took only 54 seconds. To use a popular cliche, it was awesome. I was stunned. "Wow! This puppy is fast," I muttered, sinking into verbal oblivion.

I ran Norton's Computing Index; we'll skip the arguments about the merits of various benchmarks. My old PC got a score of 1, of course. The AST Premium 286 I use at my full-time client site scored an 11. My new 386 turned in a score of 22. The disk drives also turned in a decent score, and the result is a system that can really whip through the compile process. This is what I bought the faster processor for.

I've been telling clients for some time that they really don't need a super heavy machine to do word processing; you can type only so fast. That seems to be true; I've found that the AST 286 feels really snappy, but it doesn't greatly speed up my throughput. I did notice, though, that the 286 machine made spelling checks a lot easier to take. On the 386 machine not only is spell checking faster yet, but the 386 also allows me to move around in large documents (say, 60 pages or more) as if they were only a few pages long. That makes a real difference when I'm trying to edit a long report and the midnight oil is burning.

So the new machine is *well* worth the trouble it's already caused me in routine installation headaches, and I'd certainly recommend if you can find a buyer for your old 8086 or 80286 machine, that you consider moving up to one.

A Printer, Too

Now, let's talk about the new laser printer that arrived a few weeks before I replaced the rest of the system. I bought my first laser not long after HP competition began to appear; early enough in fact that I bet my money that Canon would gain enough market share to give them wide software support. Wrong, moose breath. It was a nice printer, but there were almost no drivers for it. I learned an awful lot about font tables and laser codes, though, during the several years I owned it. The purpose of the laser was to beef up the appearance of my user manuals and reports, and with a lot of work it certainly did that. [Readers who go back as far as the November/December 1986, issue of *PJ* will remember Paul's tussle with the Canon A1.—*Ed.*]

I mentioned to one of my clients that I might be willing to part with my old printer, and the custom WordPerfect drivers to make it work, and he took me up on it. Great.

From my experience with the Canon and with HPs at client sites, I had already decided I wanted a lot more flexibility in font selection. I can't tell you how many times I needed the same typeface in a different size and had to settle for something else just because the size was right. Bah.

PostScript seems to be getting the upper hand in page description languages. As I get older (and maybe just a little wiser), I find I get more conservative about taking unknown paths. If PostScript is winning, why not go with it? Their standard package, available on a number of laser printers, offers 35 fonts in virtually any point size.

It used to take me about 20 minutes . . . . I tried the same thing with the new system, and it took only 54 seconds.

I looked around at lasers and decided that although the Apple LaserWriter seemed to have all the market share, I didn't want to pay that sort of money. Plenty of new printers were coming out soon for a little less, and I figured any machine with PostScript would perform about the same; I wasn't too concerned about white-writing versus black-writing engines and all that although getting sheets out in the proper order and having a tray that holds more than two sheets of paper were pluses. When AST announced their TurboLaser/PS for $3,995 I decided that that machine would do nicely.

The first problem was that a fellow ("I can get it for you wholesale") consultant who set up my printer deal couldn't fit the thing in his car. The AST laser is typical of the new large tray models; we're talking a big piece of equipment. I happen to drive a very large car with a trunk big enough to hold a small imported car instead of a spare tire, so hauling the printer wasn't a big chore. Getting it out of the trunk, though, wasn't easy. I could have used one of those engine hoists that shade tree mechanics always have around.

Set up was remarkably easy. The boxes inside the outer box were well marked, and everything fit together nicely. I had had occasion to fool around with a Ricoh engine on a client's laser, so the odd little bottle to collect spilled toner seemed perfectly natural, as did the procedure for dumping in two cartridges of the nasty black stuff. That was easy.

The next nice part was turning it on, when after warming up, it spit out an attractive test sheet with a sample of the fonts and the notice that this was the ninth PostScript page to come out of the printer. Great.

The bad part came when I tried to actually use the printer for something—you know, something simple—a little screen dump of a directory just to try it out. No dice. Nothing. Nada. It lit up and took data; then after awhile it just dumped the data into the bit bucket in the sky.

After reading the remarkably skimpy manual several times, I concluded that the TurboLaser/PS is a real nice PostScript printer, but that it can't do anything else—just printing good old ASCII characters sent to it, for instance. The AST manual men-

What have YOU been missing?

C Programmer's Toolbox Volumes I & II

CFlow™: Determine program hierarchy, external R/T library functions, etc.;
CLint™: More rigorously check program syntax;
CPrint™: Beautify source programs to user selected formats;
CXref™: Cross reference and check symbol usage;
CritPath™: Determine a program's critical path;
PMon™: Monitor program/OS execution; and more...

Create better, faster, higher quality and easier to read programs in a fraction of the time. Let your system do the work for you. With 23 powerful, state of the art tools for the IBM PC and compatibles, both beginners and experts will find programming a breeze.

Easy to use. Unlimited program sizes. Online documentation...

At $79.95 per volume or $130 for both with a 30 day money back guarantee, the Toolbox is simply the best value today. The Toolbox works with your existing C compiler(s) and enhances your development environment.

Call and Order Today
Visa, MasterCard Accepted

™

MMC AD Systems
Box 360845 Milpitas, California 95035
(408) 263-0781
"The C Tool Specialists"

Number 94 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

tioned a PostScript reference manual, but of course it wasn't included with the system. Hey, I know they had to find some way to cut out a few bucks, but making a printer that understands only one "interesting" language and then not including any documentation about that language seems crude. I've seen other brands that were shipped with PostScript reference books.

Oh yeah, the manual says you can buy an option so that the printer will emulate something else. More about emulation in a moment.

I mention all this because there are some interesting ramifications in choosing a PostScript printer, things you ought to know as a well-read consultant. First, if you use WordPerfect and ever get into this sort of predicament with *any* new printer, try one of WordPerfect's drivers. If you understand how *their* configuration system works, by examining their drivers you can learn an awful lot about a particular printer, such as how *it* really works. I got a little excited about installing the printer so that I could get anything at all to print on it, and when I looked in WordPerfect's drivers for AST, I saw a Dutch and a Swiss PS (PostScript) driver.

Super! I was surprised that WordPerfect was so far ahead of things that they managed to get drivers in there for a printer that wasn't invented when my copy of the word processing package was shipped. Silly me. Neither Dutch nor Swiss worked. A call to WordPerfect revealed that I had to install the AST laser as an Apple LaserWriter. "Then why the hell did you label those AST drivers as PS?" I asked. The nice lady explained that PS means "Proportional Spacing." Oops, sorry, Lady. Sure enough, the LaserWriter driver works fine.

Now, the second thing you should know about all this is that WordPerfect will allow you to set up only eight fonts. That means you get a series of say, Times Roman, fonts in various sizes, all in portrait orientation. You need to install a different driver to get the same thing in landscape. Okay, so you just install a different driver for each of several different printers you've defined and then change printers in the software if you want different fonts.

Unfortunately, that means you can't really select fonts very easily since you can't mix fonts from different printer definitions without customizing the definitions. After all, you need a character table to tell the system how wide each of the characters is. For the record, the Times Roman driver provides regular Times Roman, italic, bold, and bold-italic in 12 point; regular and italic in 9 point; regular in 5 point; and bold in 18 point. (A point is one 72nd of an inch; 12 point is normal text, 9 is a little small, 5 is the proverbial fine print, and 18 point makes nice headlines.)

The third thing you should know about all this is that PostScript drivers may require an *awful* lot of set up. The WordPerfect driver calls a separate file at initialization; you need those files provided on the printer disks with "PS" extensions. I looked in the file and was properly astounded. There's an awful lot of stuff in there. WordPerfect identifies all 35 fonts even though they're not used, and with a little poking around in there, you can easily see how fonts are called out in the language. In about ten minutes of mucking around, I created some huge letters for signs. Of course, I wouldn't have been able to write a book that way since without a proper character table WordPerfect would not have known how to break lines and so forth. Still, creating perfectly formed letters two inches high can feed your megalomania.

Consequences

So what does all this mean? It means that if you're designing software and think you might like it to run on PostScript printers one day, you'd better rethink your user definable printer menus. Your software is not only going to have to be able to send customizable printer codes, but it's also probably going to have to be able to send an entire initialization file. Scary thought, huh? Once I accepted the fact that an entire external file might be sent, I realized how this technique opens up a system. It doesn't matter how complex the initialization is; just cram it all in a file.

On the plus side, think what you can do with all the features! By the way, what I'm seeing in some applications software today is an explicit configuration question about whether a PostScript printer is attached. My guess is that all the printing branches to one of two routines: one for all the regular printers in the world and one for the PostScript machines.

Now back to emulation. I called AST. I figured they'd probably hold me up for another hundred bucks or so to get the printer functioning normally. They told me they have an EL option that gives you HP LaserJet Plus emulation, Diablo emulation, and eight standard fonts. I didn't need *that much* emulation, but that's the way it comes. It was a good thing I was sitting down when I asked them how much the EL option lists for. They told me $695!

I didn't want to point out to AST that their inexpensive PostScript printer had suddenly become less a bargain because I was too busy enjoying the excellent quality of output—and the fact that I can now rotate fonts.

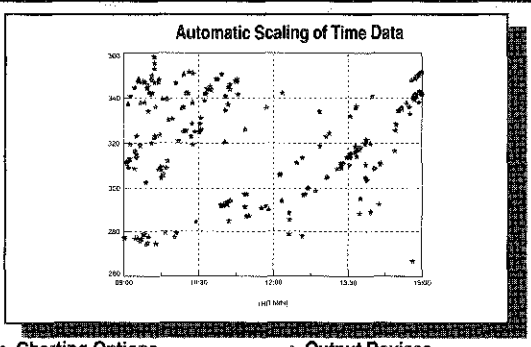
Like most lasers, the Turbo/PS doesn't handle envelopes very well. I find it easier to print the address sideways on a blank page, throw it in my copier, and feed an envelope through the copier. This sounds much worse than it really is; it's a fact of life with most lasers, though.

The Bottom Line

It's difficult to come up with quantitative data to support the acquisition of new tools. After the learning period is over, new tools, if wisely chosen, can provide continuing benefits that result in higher quality work at less cost.

Continued...

MIGHTYSOFT™ Introduces
G-WIZ® Business & Scientific Graphics
G-RUN™ Module for Integrated Database Graphics



- **Charting Options**
  Page Position
  Chart Size
  Marker Size & Style
  Show Missing Values
  Line Thickness
  Line Style
  23+ Fonts
  Full Color Support
  32 Fill Patterns
- **Power and Ease**
  Unlimited Datapoints
  In-context Help
  Menu-Driven
- **Output Devices**
  CGA, EGA, VGA
  Hercules
  Color Inkjet Printers
  HP Plotters
  Laserjet Printers
  Postscript Printers
  Slidemakers
  Metafiles for Desktop Publishing
  Toshiba Printers
  Epson Series Printers

  Over 250 supported at maximum device resolution!
- **Charts Supported**
  Horizontal Stacked Bar
  Horizontal Clustered Bar
  Vertical Stacked Bat
  Vertical Clustered Bar
  Line
  Scatter (X,Y)
  Logarithmic Curve Fit
  Exponential Curve Fit
  Straight Line Regression
  Prediction Intervals
  Confidence Intervals
  Moving Average
  Frequency Histogram
  Pie
  Area Fill
- **Data Types**
  Numbers
  Text
  Time
  Dates
  Months
  Quarters
- **Scaling Features**
  Manual
  Semi-Automatic
  Automatic
  Linear, Log, Semi-Log

• G-WIZ Graphics—$395 | • G-RUN Database Module—$195

P.O. Box 51048, Seattle, WA 98115. Phone: (206) 525-3766. Telex: 6503323229.

Number **179** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FINALLY!

EASYFLOW

An on-screen flowchart processor that knows about flowcharts - not just another "screen draw" program that makes you do most of the work. EASYFLOW is a powerful full-screen graphics program dedicated to flowcharts and organization charts. With it you can quickly compose charts. More important, you can easily modify charts so they are always up to date.

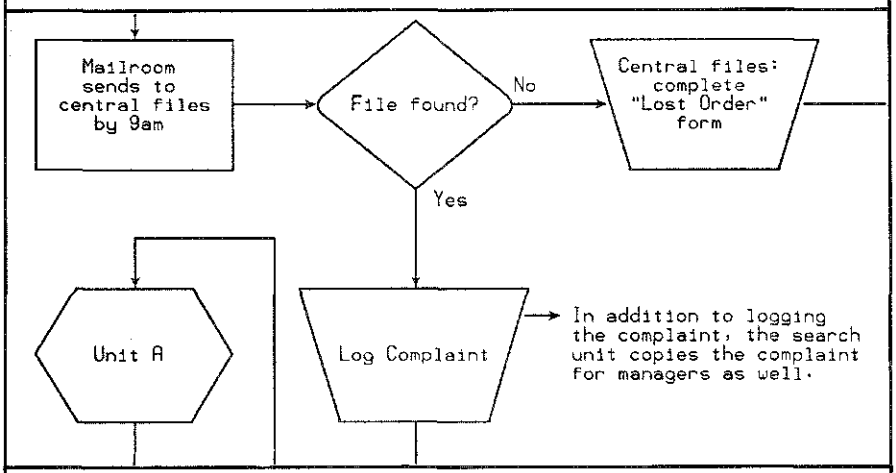
- **Automatic:** Fully automatic text centering within shapes, both horizontally and vertically. Fully automatic line routing & re-routing.
- **Fast:** Written in assembly language for speed.
- **Large:** Charts up to 417 columns wide by 225 lines high. Chart too large for your printer? *EASYFLOW* automatically breaks the chart up & prints it in page size pieces.
- **Standard:** All standard flowcharting shapes included. Custom shapes can be ordered.
- **User friendly:** Don't take our word for it. PC Magazine* says "*EASYFLOW* lives up to its name. It's hard to imagine any easier and more flexible way to produce basic and even complex flowcharts".
- **Mouse:** Optional but fully supported.
- **It prints:** On most popular matrix printers including IBM, Epson, Toshiba, HP LaserJet, LaserJet-Plus and many others.
- **It plots:** On HP7440, 7475, 7550, 7570, 7585B and compatible plotters.
- **It works:** We are contractually prevented from mentioning the name of the "big eight" accounting firm that purchased a world-wide site license, but we can tell you that they spent months evaluating all available flowcharting packages before choosing *EASYFLOW*.
- **Rush delivery:** Order by noon today (eastern time) and we'll have it to you by courier tomorrow**. Rush delivery charge is $15.00 (instead of $2.00) and is available only in USA & Canada.
- **Documented:** 100 page manual plus over 150 screens of context sensitive help.

EASYFLOW works on IBM PC's, IBM PS/2 and compatibles. Requires 384 K memory, DOS 2.0 or higher and an IBM CGA/EGA/VGA or Hercules monochrome compatible adapter card.

Order direct for only **$149.95** + $2.00 S&H (USA/Canada), $10.00 (foreign). Payment by M.O., cheque, VISA, Mastercard or Company PO.

* March 10, 1987 issue, page 278.

** Rush orders are shipped by Purolator Courier and normally arrive the next business day to most locations. Remote destinations take longer.


Mailroom sends to central files by 9am
File found?
No
Central files: complete "Lost Order" form
Yes
Unit A
Log Complaint
In addition to logging the complaint, the search unit copies the complaint for managers as well.

The chart fragment above was produced on an HP LaserJet-Plus and is actual size and unretouched. Publication quality charts like this can be produced using only minutes of preparation time and seconds of print time.

HavenTree Software Limited
PO Box 1093-V
Thousand Island Park, NY 13692

Order Desk: 1-800-267-0668
Information: (613) 544-6035 ext 31
Telefax(G3): (613) 544-9632

Number **154** on the Reader Service Card

In addition, good tools are salubrious; they make you feel good.

Evaluate your own practice. What aspects seem inefficient to you? How can you automate those tasks? How much will it cost to do so? Make sure you can afford the new tools regardless of how cost effective they'll be. If you're really thorough, you'll spend days creating spreadsheets that demonstrate conclusively how you'll benefit from the new technology and how it will pay for itself in only 3.1415 months; otherwise, if you can afford it and it looks as if it will save you some time (and you can't stop talking about it anyway), then go ahead and do it. Just pick a time when you don't have a big backlog of critical deliverables. Oh yes. If you have a significant other, better get him or her in the mood for the new equipment, too; you might wind up using your new hardware and software inside a tent in the backyard. ♦

Teamwork, $995
Cadre Technologies, Inc.
222 Richmond Street
Providence, RI 02903
(401) 351-CASE

Send your comments or questions to

Paul A. Barkley
The Barbican Company
6845 Elm Street
Penthouse One
McLean, VA 22101

Microsoft®
QuickC™ / C5.0
Graphics Library

NOW, A LINK!

Hercules™
InColor Card
Graphics Card Plus
Graphics Card

$34.95 Postpaid

Soft Evolutions, Inc.
484 Lake Park Ave., #17
Oakland, CA 94610-2730
(415) 523-8247

Number **177** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

OMNIVIEW

ROCK SOLID MULTITASKING!

Packed with the power you need, OMNIVIEW takes you beyond the limits of DOS.

- Time-slices multiple jobs
- Takes full advantage of EMS 4.0
- Optional support for '386 Systems
- Manages megabytes of programs
- Controls resident utilities
- Lets you cut and paste between programs

OMNIVIEW unleashes your PC & converts it into an easy-to-use, multitasking workstation. Whether you have an 80386 or 8088 based computer, OMNIVIEW will squeeze out every ounce of power.

Solid & bulletproof, it's 47K assembly language engine gives you all the power without added overhead or complexity. Just ask the reviewers.*

"Sunny Hill provides true multitasking for most PC applications. It performs remarkably well, causing almost no noticeable reduction in the performance of each program . . . The user manual is well organzied and easy to read . . . It is an excellent product."

PC Tech Journal Sept '87

"It does an amazing job of handling not only multitasking, but in many cases, concurrent processing as well. I was amazed at it's success in handling the "impossible" combinations I threw at it."

"It is one of the best designed & implemented programs I've ever worked with. It is also one of the most bulletproof."

"What sets it apart from the pack is the fact that it runs in a network environment."

Data Based Advisor Aug '87

OMNIVIEW requires an IBM PC, XT, AT or Jr compatible, and PC or MS DOS 2.x to 3.x. To get your copy, call toll free:

(800) 367-0651

or send $89.95 + 3.00 S&H ($8.00 Intl.) to:

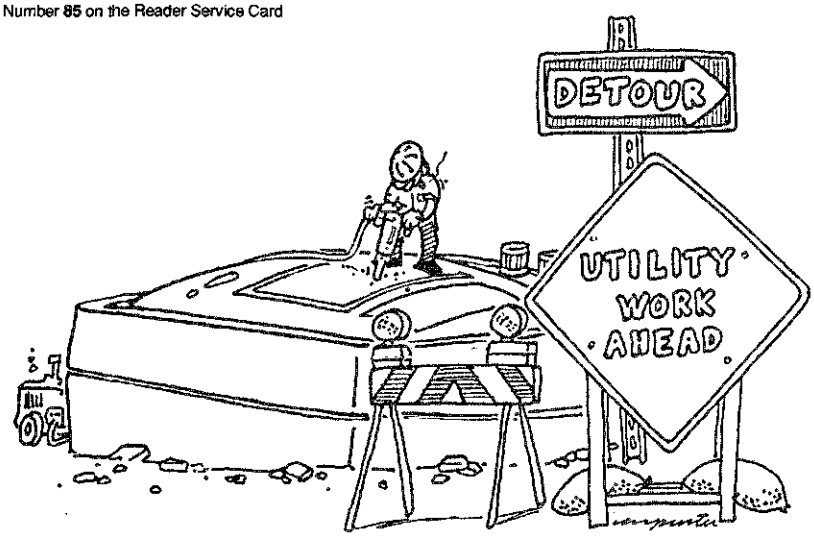
Sunny Hill Software

PO Box 55278, Seattle, WA 98155-5278

For more information call (206) 367-0650 *Omniview formerly Taskview

DoubleDOS trademark Softlogic Solutions, Concurrent PC-DOS reg. trademark Digital Research Corp., MS Windows trademark Microsoft Corp., DESQview trademark Quarterdeck Office Systems, Inc. TopView trademark IBM Corp.

Number 85 on the Reader Service Card



NROFF/PC™

The REAL *Thing for* DOS

NROFF/PC is a complete text formatting system for *MS-DOS* systems. Including:

NROFF The *powerful* UNIX text formatter

TBL A tool to assist with the layout of tabular material in *Nroff* documents

MM A comprehensive *Nroff* macro package for preparing books and technical manuals

NEQN A tool for describing mathematical equations in *Nroff* documents

- All tools are a complete port from the AT&T Documentor's Workbench 2.0
- It's **Fast!** We've modified *Nroff* especially for DOS for lightning speed
- Supports any Dot Matrix printer and many laser printers
- Specially Priced At **$99**
- A complete *Troff* typesetting system is available **NOW** for LaserJet and PostScript printers on **MS-DOS** for *$695*, XENIX and Microport UNIX for *$795*.

Elan

Elan Computer Group, Inc.
410 Cambridge Ave., Suite A
Palo Alto, CA 94306
(415) 322-2450

Visa and MasterCard Accepted

Unix is a tradmark of AT&T
MS-DOS and Xenix are tradmarks of Microsoft

Number 173 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

OS/2 Memory Management

excerpted from *Inside OS/2*

by
Gordon Letwin

Our thanks go to Microsoft's Chief Architect for Systems Software Gordon Letwin and Microsoft Press for permission to reprint the following excerpts from the Architecture section of Inside OS/2.

A primary function of any multitasking operating system is to allocate system resources to each process according to its need. The scheduler allocates CPU time among processes; the *memory manager* allocates both physical memory and virtual memory.OS/2 provides an extensive memory management API. These excerpts focus on areas that may not be completely self-explanatory.

Shared Memory

OS/2 supports two kinds of shared memory—*named* shared memory and *giveaway* shared memory. In both, the memory object shared is a segment. Only an entire segment can be shared; sharing part of a segment is not possible. Named shared memory is volatile because neither the name of the named shared memory nor the memory itself can exist on the FAT file system. When the number of processes using a shared memory segment goes to zero, the memory is released. Shared memory can't stay around in the absence of client processes; it must be reinitialized via **DosAllocShrSeg** after a period of nonuse.

Giveaway shared memory allows processes to share access to the same segment. Giveaway shared memory segments don't have names because processes can't ask to have access to them; a current user of the segment has to give access to the segment to a new client process. The term *giveaway* is a bit of a misnomer because the giving process retains access to the memory—the access is "given" but not especially "away." Giveaway shared memory is not as convenient as named shared memory. The owner (one of the owners—those with access to a giveaway shared segment can also give it away) has to know the Process ID (PID) of the recipient and then communicate the recipient's segment selector (returned by **DosGiveSeg**) to that recipient process via some form of InterProcess Communication (IPC)).

Despite its limitations, giveaway shared memory has important virtues. It's a fast and efficient way for one process to transfer data to another; and because access is passed "hand to hand," the wrong process cannot accidentally or deliberately gain access to the segment. Most clients of giveaway shared memory don't retain access to the segment once they've passed it off; they typically call **DosFreeSeg** on their handle after they've called **DosGiveSeg**. For example, consider the design of a database dynlink subsystem that acts as a front end for a database serving process. As part of the dynlink initialization process, the package arranged for its client process to share a small named shared memory segment with the database process. It might be best to use a named pipe or named shared memory—created by the database process—to establish initial communication and then use this interface only to set up a private piece of giveaway shared memory for all further transactions between the client process (via the dynlink subsystem) and the database process. Doing it this way, rather than having one named shared segment hold service requests from all clients, provides greater security. Because each client has its own separate shared memory communications area, an amok client can't damage the communications of other clients.

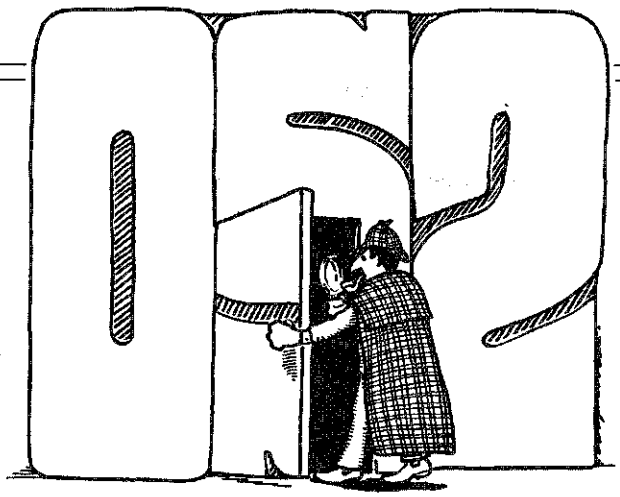
When a client process asks the database process to read it a record, the database process must use a form of IPC to transfer the data to the client. Pipes are too slow for the volume of data that our example anticipates; shared memory is the best technique. If we were to use named shared memory, the database package would have to create a unique shared memory name for each record, allocate the memory, and then communicate the name to the client (actually, to the dynlink subsystem called by the client) so that it can request access. This process has some drawbacks:

- A new unique shared memory name must be created for each request. We could reuse a single shared memory segment, but this would force the client to copy the data out of the segment before it could make another request—too costly a process for an application that must handle a high volume of data.
- Creating named shared memory segments is generally slower than creating giveaway shared memory segments, especially if a large number of named shared memory objects exist, as would be the case in this scenario. The client spends more time when it then requests access to the segment. Creating named shared memory segments is plenty fast enough when it's done once in a while, but in a high-frequency application such as our example, it could become a bottleneck.

Instead, the database process can create a giveaway shared memory segment, load the data into it, and then give it to the

Reprinted by Permission of Microsoft Press.
Copyright © 1988 by Microsoft Press. All Rights Reserved

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



client process. The database process can easily learn the client's PID; the dynlink interface, which runs as the client process, can include it as part of the data request. Likewise, the database process can easily return the new client selector to the client. This process is fast and efficient and doesn't bog down the system by forcing it to deal with a great many name strings.

Note that you must specify, at the time of the **DosAllocSeg**, that the segment might be "given away." Doing so allows OS/2 to allocate the selector in the disjoint space.

Huge Memory

The design of the 80286 microprocessor specifies the maximum size of a memory segment as 64 KB. For many programs, this number is far too small. For example, the internal representation of a large spreadsheet commonly takes up 256 KB or more. OS/2 can do nothing to set up a segment that is truly larger than 64 KB, but the OS/2 facility called *huge segments* provides a reasonable emulation of segments larger than 64 KB. The trick is that a huge segment of, for example, 200 KB is not a single segment but a group of four segments, three of which are 64 KB and a fourth of 8 KB. With minimal programming burden, OS/2 allows an application to treat the group of four segments as a single huge segment.

When a process calls **DosAllocHuge** to allocate a huge segment, OS/2 allocates several physical segments, the sum of whose size equals the size of the virtual huge segment. All component segments are 64 KB, except possibly the last one. Unlike an arbitrary collection of segment selectors, **DosAllocHuge** guarantees that the segment selectors it returns are spaced uniformly from each other. The selector of the N+1th component segment is that of the Nth segment plus i, where i is a power of two. The value of i is constant for any given execution of OS/2, but it may vary between releases of OS/2 or as a result of internal configuration during bootup. In other words, a program must learn the factor i every time it executes; it must not hard code the value. There are three ways to learn this value. First, a program can call **DosGetHugeShift**; second, it can read this value from the global infoseg; and third, it can reference this value as the undefined absolute externals DOSHUGESHIFT ($\log_2(i)$) or DOSHUGEINCR (i). OS/2 will insert the proper value for these externals at loadtime. This last method is the most efficient and is recommended. Family API programs should call **DosGetHugeShift**. Figure 1 illustrates the layout of a 200 KB huge memory object. Selectors n + 4i and n + 5i

Protection Model

Although MS-DOS provided a simple form of memory management, OS/2 provides memory protection. Under MS-DOS 3.x, for example, a program should ask the operating system to allocate a memory area before the program uses it. Under OS/2, a program *must* ask the operating system to allocate a memory area before the program uses it. The 80286 microprocessor contains special memory protection hardware. Each memory reference that a program makes explicitly or implicitly references a segment selector. The segment selector, in turn, references an entry in the Global Descriptor Table (GDT) or the Local Descriptor Table (LDT), depending on the form of the selector. Before any program, including OS/2 itself, can reference a memory location, that memory location must be described in an LDT or a GDT entry, and the selector for that entry must be loaded into one of the four segment registers.

This hardware design places some restrictions on how programs can use addresses:

- A program cannot address memory not set up for it in the LDT or GDT. The only way to address memory is via the LDT and GDT.
- Each segment descriptor in the LDT and GDT contains the physical address and the length of that segment. A program cannot reference an offset into a segment beyond that segment's length.
- A program can't put garbage (arbitrary values) into a segment register. Each time a segment register is loaded, the hardware examines the corresponding LDT and GDT to see if the entry is valid. If a program puts an arbitrary value—for example, the lower half of a floating point number—into a segment register, the arbitrary value will probably point to an invalid LDT or GDT entry, causing a General Protection (GP) fault.
- A program can't execute instructions from within a data segment. Attempting to load a data segment selector into the CS register (usually via a far call or a far jump) causes a GP fault.
- A program can't write into a code segment. Attempting to do so causes a GP fault.
- A program can't perform segment arithmetic. Segment arithmetic refers to activities made possible by the addressing mechanism of the 8086 and 8088 microprocessors. Although they are described as having a segment architecture, they are actually linear address space machines that use offset registers—the so-called segment registers. An 8086 can address 1 MB of memory, which requires a 20-bit address. The processor creates this address by multiplying the 16-bit segment value by 16 and adding it to the 16-bit offset value. The result is an address between 0 and 1,048,575 (that is, 1 MB). (Actually, it's possible to produce addresses beyond 1 MB (2^{20}) by this method if a large enough segment and offset value are chosen. The 8086 ignores the carry into the nonexistent 21st address bit, effectively wrapping around such large addresses into the first 65 KB–16 bytes of physical memory.) The reason these are not true segments is that they don't have any associated length and their names (that is, their selectors) aren't names at all but physical addresses divided by 16. These segment values are actually scaled offsets. An address that has a segment value of 100 and an offset value of 100

Continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

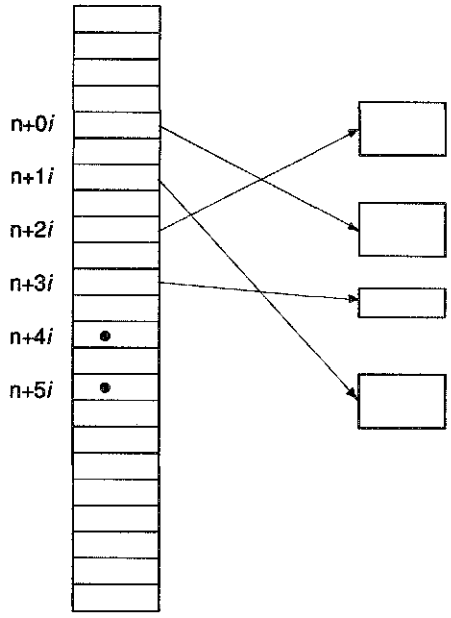


Figure 1. Huge memory objects.

are currently invalid but are reserved for future growth of the huge object.

Once an application has the first segment selector of the huge segment group, called the *base segment*, and the value $\log_2(i)$, computing the address of the Nth byte in the huge segment is easy. Take the high-order word of the value of N (that is, N/64 KB), shift it left by $\log_2(i)$ (that is, by the **DosHugeShift** value), and add the base segment selector returned by **DosAllocHuge**. The resultant value is the segment selector for the proper component segment; the low-order 16 bits of *i* are the offset into that segment. This computation is reasonably quick to perform since it involves only a shift and an addition.

Huge segments can be shrunk or grown via **DosReallocHuge**. If the huge segment is to be grown, creating more component physical segments may be necessary. Because the address generation rules dictate which selector this new segment may have, growing the huge segment may not be possible if that selector has already been allocated for another purpose. **DosAllocHuge** takes a maximum growth parameter; it uses this value to reserve sufficient selectors to allow the huge segment to grow that big. Applications should not provide an unrealistically large number for this argument because doing so will waste Local Descriptor Table (LDT) selectors.

The astute reader will notice that the segment arithmetic of the 8086 environment is not dead; in a sense, it's been resurrected by the huge segment mechanism. Applications written for the 8086 frequently use this technique to address memory regions greater than 64 KB, using a shift value of 12. In other words, if you add $2^{\wedge 12}$ to an 8086 segment register value, the segment register will point to an address $2^{\wedge 12}*16$, or 64 KB, further in physical memory. The offset value between the component segment values was always 4096 because of the way the 8086 generated addresses. Although the steps involved in computing the segment value are the same in protect mode, what's actually happening is considerably different. When you do this computation in protect mode, the segment selector value has no inherent relationship to the other selectors that make up the huge object. The trick works only because OS/2 has arranged for equally spaced-out selectors to exist and for each to point to an area of physical memory of the appropriate size. Figure 2 illustrates the similarities and differences between huge model addressing in real and protect modes. The application code sequence is identical: A segment selector is computed by adding *N*i* to

(shown as 100_{10}:100_{10}) and the address (99_{10}:116_{10}) both refer to the same memory location.

Many real mode programs take advantage of this situation. Some programs that keep a great many pointers store them as 20-bit values, decomposing those values into the segment:offset form only when they need to de-reference the pointer. To ensure that certain objects have a specific offset value, other programs choose a matching segment value so that the resultant 20-bit address is correct. Neither technique works in OS/2 protect mode. Each segment selector describes its own segment, a segment with a length and an address that are independent of the numeric value of the segment selector. The memory described by segment N has nothing in common with the memory described by segment N+4 or by any other segment unless OS/2 explicitly sets it up that way.

The segmentation and protection hardware allows OS/2 to impose further restrictions on processes:

- Processes cannot edit or examine the contents of the LDT or the GDT. OS/2 simply declines to build an LDT or GDT selector that a process can use to access the contents of those tables. Certain LDT and GDT selectors describe the contents of those tables themselves, but OS/2 sets them up so that they can be used only by ring 0 (that is, privileged) code.
- Processes cannot hook interrupt vectors. MS-DOS version 3.x programs commonly hook interrupt vectors by replacing the address of the interrupt handler with an address from their own code. Thus, these programs can monitor or intercept system calls made via INT 21h, BIOS calls also made via interrupts, and hardware interrupts such as the keyboard and the system clock. OS/2 programs cannot do this. OS/2 declines to set up a segment selector that processes can use to address the interrupt vector table.
- Processes cannot call the ROM BIOS code because no selector addresses the ROM BIOS code. Even if such a selector were available, it would be of little use. The ROM BIOS is coded for real mode execution and performs segment arithmetic operations that are no longer legal. If OS/2 provided a ROM BIOS selector, calls to the ROM BIOS would usually generate GP faults.
- Finally, processes cannot run in ring 0, that is, in privileged mode. Both OS/2 and the 80286 hardware are designed to prevent an application program from ever executing in ring 0. Code running in ring 0 can manipulate the LDT and GDT tables as well as other hardware protection features. If OS/2 allowed processes to run in ring 0, the system could never be stable or secure. OS/2 obtains its privileged (literally) state by being the first code loaded at boot time. The boot process takes place in ring 0 and grants ring 0 permission to OS/2 by transferring control to OS/2 while remaining in ring 0. OS/2 does not, naturally, extend this favor to the application programs it loads; it ensures that applications can run only in ring 3 user mode. (Applications can also run in ring 2.) ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

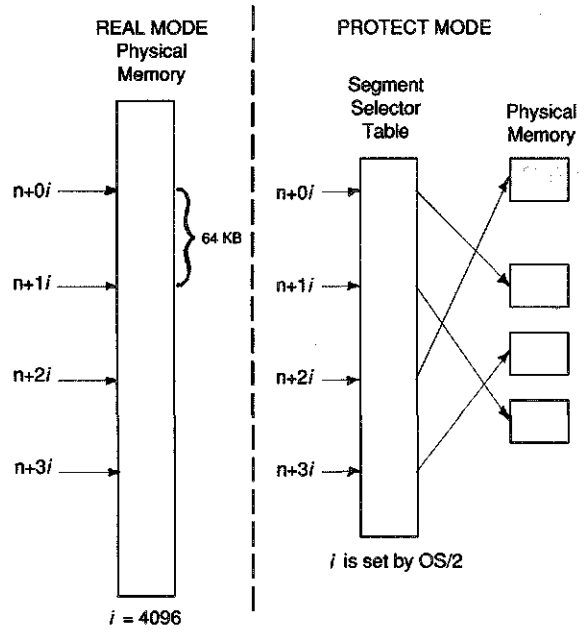


Figure 2. Huge model addressing in real mode and in protect mode.

the base selector. In real mode *i* is always 4096; in protect mode OS/2 provides *i*.

Although the similarity between 8086 segment arithmetic and OS/2 huge segments is only apparent, it does make it easy to write a program as a *dual mode* application. By using the shift value of 12 in real mode and using the OS/2 supplied value in protect mode, the same code functions correctly in either mode.

Executing from Data Segments

We saw that OS/2 provides the huge segment mechanism to get around the segment size restriction imposed by the hardware. OS/2 likewise circumvents another hardware restriction—the inability to execute code from data segments. Although the demand loading, the discarding, and the swapping of code segments make one use of running code from data segments—code overlays—obsolete, the capability is still needed. Some high-performance programs—the presentation manager, for example—compile "on the fly" special code to perform time-critical tasks, such as flipping bits in EGA display memory. The optimal sequence may differ depending on several factors, so a program may need to compile such code and execute it, gaining a significant increase in efficiency over some other approach. OS/2 supports this need by means of the **DosCreateCS-Alias** call.

When **DosCreateCSAlias** is called with a selector for a data segment, it creates a totally different code segment selector (in the eyes of 80286 hardware) that by some strange coincidence points to exactly the same memory locations as does the data segment selector. As a result, code is not actually executing from a data segment but from a code segment. Because the code segment exactly overlaps that other data segment, the desired effect is achieved. The programmer need only be careful to use the data selector when writing the segment and to use the code selector when executing it.

Memory Suballocation

All memory objects discussed so far have been segments. OS/2 provides a facility called *memory suballocation* that allocates pieces of memory from within an application's segment. Pieces of memory can be suballocated from within a segment, grown, shrunk, and released. OS/2 uses a classic heap algorithm to do this. The **DosSubAlloc** call uses space made available from earlier **DosSubFrees** when possible, growing the segment as necessary when the free heap space is insufficient. We will call the pieces of memory returned by **DosSubAlloc** *heap objects*.

The memory suballocation package works within the domain of a process. The suballocation package doesn't allocate the memory from some "system pool" outside the process's address space, as does the segment allocator. The suballocation package doesn't even allocate segments; it manages only segments supplied by and owned by (or at least accessible to) the caller. This is a feature because memory protection is on a per-segment basis. If the suballocation package were to get its space from some system global segment, a process that overwrote its heap object could damage one belonging to another process. Figure 3 illustrates memory suballocation. It shows a segment j being suballocated. H is the suballocation header; the shaded areas are free space.

We said that the memory suballocator subdivides segments that are accessible to the client process. This means that you can use it to subdivide space in a shared memory segment. Such a technique can be handy when two or more processes are using a shared memory segment for intercommunication, but there is risk because an error in one process can easily corrupt the heap objects of another.

The memory suballocation package released with OS/2 version 1.0 doesn't reside in the kernel; it's effectively a subroutine package. OS/2 in an 80286 environment will undoubtedly preserve this approach in future releases, but a forthcoming 80386 version of OS/2 may not. The 80386 architecture supports paged virtual memory, so memory swapping (actually, paging) can take place on part of a segment. This future paging environment may precipitate some changes in the memory suballocator. Perhaps we'll want to rearrange the heap for better efficiency with paging, or perhaps the OS/2 kernel will want to become involved so that it can better anticipate paging demands. In any case, any future release of OS/2 has complete flexibility to upgrade the memory suballocation package in any externally compatible fashion, thanks to the standard interface provided by dynamic links.

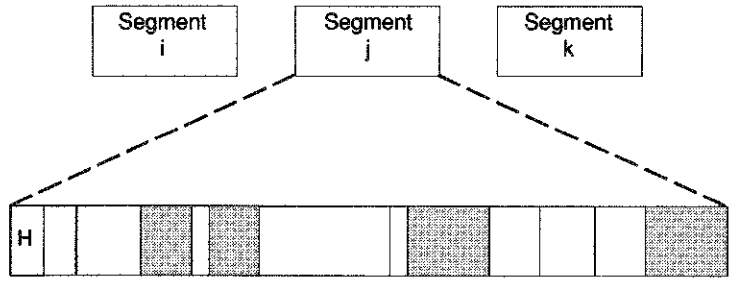


Figure 3. Memory suballocation.

Continued...

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Segment Swapping

One of the most important features of the 80286 memory management hardware is swapping support. Swapping is a technique by which some code or data segments in memory are written to a disk file, thus allowing the memory they were using to be reclaimed for another purpose. Later, the swapped-out code or data is reloaded into memory. This technique lets you run more programs than can simultaneously fit in memory; all you need is enough memory to hold the programs that are running at that particular moment. Quiescent programs can be swapped to disk to make room for active ones. Later, when the swapped programs become active, OS/2 reads them in and resumes them. If necessary OS/2 first makes memory available by swapping out another quiescent program.

Although I used the word *program* above, swapping is actually done on a segment basis. Segments are swapped out individually and completely; the OS/2 swapping code doesn't pay attention to relationships between segments (they aren't swapped in groups), and the 80286 hardware does not allow only part of a segment to be swapped. I simplified the concept a bit in the above paragraph. You need not swap out an entire process; you can swap out some segments and leave others in memory. OS/2 can and commonly does run a process when some of its segments are swapped out. As long as a process does not try to use the swapped-out segments, it runs unhindered. If a process references a swapped-out segment, the 80286 hardware generates a special trap that OS/2 intercepts. The *segment fault* trap handler swaps in the missing segment, first swapping out some other if need be, and then the process resumes where it left off. Segment faulting is invisible to a process; the process executes normally, except that a segment load instruction takes on the order of 30 milliseconds instead of the usual 3 microseconds.

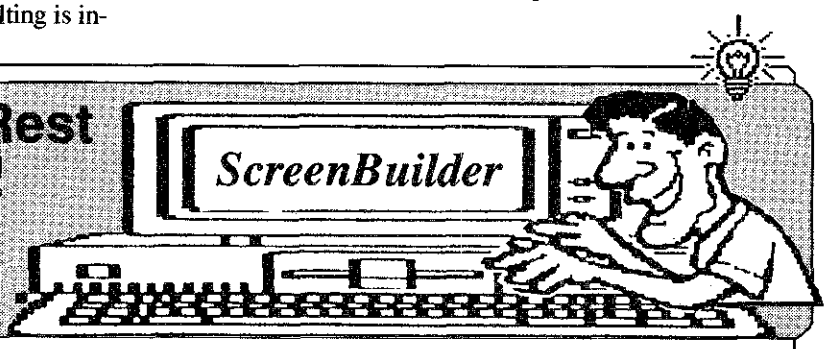
When memory is depleted and a segment must be swapped, OS/2 has to choose one to swap out. Making the right choice is important; for example, consider a process that alternates references between segment A and segment B. If A is swapped out, a poorly designed system might choose B to swap out to make room for A. After a few instructions are executed, B has to be swapped in. If A is in turn swapped out to make room for B, the system would soon spend all its time swapping A and B to and from the disk. This is called *thrashing*, and thrashing can destroy system performance. In other words, the effect of swapping is to make some segment loads take 10,000 times longer than they would if the segment were in memory. Although the number 10,000 seems very large, the actual time of about 30 milliseconds is not, *as long as we don't have to pay those 30 milliseconds very often.*

A lot hinges on choosing segments to swap out that won't be referenced in the near future. OS/2 uses the Least Recently Used (LRU) scheme to determine which segment it will swap out. The ideal choice is the segment—among those currently in memory—that will be referenced last because this postpones the swap-in of that segment as long as possible. Unfortunately, it's mathematically provable that no operating system can predict the behavior of arbitrary processes. Instead, operating systems try to make an educated guess as to which segment in memory is least likely to be referenced in the immediate future. The LRU scheme is precisely that—a good guess. OS/2 figures that if a segment hasn't been used

You've Seen all the Rest Now go for the Best!

Absolutely NO Royalties!

NOW SHIPPING Ver 2.10 with Data Entry Capabilities!



- **Create Interactive, Animated, Automated:**
  - Demos
  - Tutorials
  - User/Program Shells
  - Teaching Aids
  - Children's Education
  - Continuously Running Presentations
- **Orders of Magnitude above the rest:**
  - Can generate BASIC BLOADable Screens
  - Can generate Turbo C Source for Screens
  - Can generate ANSI files.
  - Sound and Music Capabilities
  - Mathematics and String Logic
  - User Inputs and Outputs
  - Printer Control
  - Loops, Subroutines, Decision Making
  - Screens are directly usable by Application Programs
  - Supports all 256 Colors and Characters
  - Screen Capture Program
  - Runs on CGA, EGA and Monochrome
  - Much, Much More!
- **Easy to use. Has context sensitive help.**
  - Complete, comprehensive 150+ page User Manual guides you through every step of the way.

Just $59.95 Complete!

Sample Demo Disk available for $2.00 (Credited towards your first order)

Call Today to order by credit card or send $59.95 plus $3.00 shipping/handling to:

GILMORE SYSTEMS

P. O. Box 3831

Beverly Hills, CA 90212-0831

VISA **(213) 275-8006** MasterCard

Requires IBM PC, XT, AT or compatible. DOS 2.0 or higher. 320 K RAM minimum. Fixed disk recommended.

Number 135 on the Reader Service Card

PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

in a long time then it probably won't be used for a long time yet, so it swaps out the segment that was last used the longest time ago—in other words, the least recently used segment.

Of course, it's easy to construct an example where the LRU decision is the wrong one or even the worst one. The classic example is a program that references, round robin, N segments when there is room in memory for only N–1. When you attempt to make room for segment *I*, the least recently used segment will be *I*+1, which in fact is the segment that will next be used. Authors of programs that will make repetitious accesses to large bodies of data or code should study the available literature on virtual memory systems. Remember, on an 80286, OS/2 swaps only on a segment basis. A future 80386 release of OS/2 will swap, or page, on a 4 KB page basis.

The swapping algorithm is strictly LRU among all swap-eligible segments in the system. Thread/process priority is not considered; system segments that are marked swappable get no special treatment. Some system segments are marked non-swappable, however. For example, swapping out the OS/2 code that performs swap-ins would be embarrassing. Likewise, the disk driver code for the swapping disk must not be swapped out. Some kernel and device driver code is called at interrupt time; this is never swapped because of the swap-in delay and because of potential interference between the swapped-out interrupt handling code and the interrupt handling code of the disk driver that will do the swap-in. Finally, some kernel code is called in real mode in response to requests from the 3x box. No real mode code can be swapped because the processor does not support segment faults when running in real mode.

The technique of running more programs than there is RAM to hold them is called *memory overcommit*. OS/2 has to keep careful track of the degree of overcommit so that it doesn't find itself with too much of a good thing—not enough free RAM, even with swapping, to swap in a swapped-out process. Such a situation is doubly painful: Not only can the user not access or save the data that he or she has spent the last four hours working on, but OS/2 can't even tell the program what's wrong because it can't get the program into memory to run it. To prevent this, OS/2 keeps track of its commitments and overcommitments in two ways. First, before it starts a process, OS/2 ensures that there is enough swap space to run it. Second, it ensures that there is always enough available RAM to execute a swapped-out process.

At first glance, knowing if RAM is sufficient to run a process seems simple—either the process fits into memory or it doesn't. Life is a bit more complicated than that under OS/2 because the segments of a program or a dynlink library may be marked for demand loading. This means that they won't come in when the program starts executing but may be called in later. Obviously, once a program starts executing, it can make nearly unlimited demands for memory. When a program requests a memory allocation, however, OS/2 can return an error code if available memory is insufficient. The program can then deal with the problem: make do with less, refuse the user's command, and so forth.

OS/2 isn't concerned about a program's explicit memory requests because they can always be refused; the implicit memory requests are the problem—faulting in a demand load segment, for example. Not only is there no interface to give the program an error code (a demand load segment is faulted in via a "load segment register" instruction—these CPU in-

★ NOW AVAILABLE FOR C ★ QUICKBASIC ★
★ IBM BASIC/BASICA ★ TURBO PASCAL ★

The Aspen Systems Subroutine Editor (**ASE**) you can call from your programs. Design your own screen layouts - - update in several windows simultaneously. **ASE** is:

- ★ **TRANSPORTABLE** - can be configured for any keyboard and almost any computer or language under MSDOS.
- ★ **VERSATILE** - customize input for any application.
- ★ **COMPLETE** - with full screen edit capabilities and a wide variety of automatic conversions.
- ★ **EASY TO USE** - data and screen layouts described in a single map.
- ★ **FULLY DOCUMENTED** - including examples in BASIC, PASCAL, FORTRAN, C, and even COBOL.
- ★ **AFFORDABLE-**

ASE is still only..... **$99**
DEMO (See what ASE can do for you!)..... **$ 3**

The Aspen Systems Subroutine Package (**ASP**) - the companion package for **ASE** - with many functions unavailable or difficult to perform in high level languages. Like **ASE, ASP** is:

TRANSPORTABLE ★ FULLY DOCUMENTED
EFFICIENT ★ AFFORDABLE

ASP includes 150+ subroutines: conversions; sorts; string, bit, date/time functions; decimal arithmetic, and **MORE**.

ASP is still only..... **$130**

ASE and **ASP** include support and demo programs.

Prices are PPD (continental USA).
Colorado Residents add 3%.

P. O. Box 1163
Grand Junction, CO 81502
(303) 245-3262
VISA/MasterCard accepted

ASPEN SYSTEMS

CP/M and MS-DOS are trademarks of Digital Research and Microsoft, respectively.

Number 86 on the Reader Service Card

MICROCACHE™ and SPEEDPRINT™

The hottest PC utilities you can get.
Or your money back!

MICROCACHE™ monitors disk use and "learns" to anticipate your data accesses. Then, by satisfying these requests directly from RAM instead of disk, your programs run as if they are SUPERCHARGED.

- 100% Transparent Operation
- Uses any spare RAM available
- EMS/EEMS memory support
- SPEEDPRINT™ buffer option
- Requires NO program changes
- For IBM and compatibles

~~Price $89.00~~ NOW $69.00

MasterCard VISA

AZ

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085

(800) 227-0644 . . . (816) 776-2700

Number 51 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Infosegs

OS/2 takes advantage of the 80286 Local Descriptor Table (LDT) and Global Descriptor Table (GDT) architecture in providing two special segments, called *infosegs*, that contain system information. OS/2 updates these segments when changes take place, so their information is always current. One infoseg is global, and the other is local. The global infoseg contains information about the system as a whole; the local infoseg contains process specific data. Naturally, the global infoseg is read only and is shared among all processes. Local infosegs are also read only, but each process has its own.

The global infoseg contains time and date information. The local infoseg segment contains process and thread information. The information is accurate for the currently executing thread. The subscreen group value is used by the presentation manager subsystem and is not of value to applications. For more information on global and local infosegs, see the OS/2 reference manual.

♦

structions don't return error codes!), but the program may be unable to proceed without the segment. As a result, when a program is first loaded (via a **DosExecPgm** call), OS/2 sums the size of all its impure segments even if they are marked for "load on demand." The same computation is done for all the loadtime dynlink libraries it references and for all the libraries *they* reference and so on. This final number, plus the internal system per-process overhead, is the maximum *implicit* memory demand of the program. If that much free swap space is available, the program can start execution.

You have undoubtedly noticed that I said we could run the program if there was enough swap space. But a program must be in RAM to execute, so why don't we care about the amount of available RAM space? We do care. Not about the actual amount of free RAM when we start a program, but about the amount of RAM that can be made free—by swapping—if needed. If some RAM contains a swappable segment, then we *can* swap it because we set aside enough swap space for the task. Pure segments (code segments), by the way, are not normally swapped. In lieu of a swap-out, OS/2 simply discards them. When it's time to swap them in, OS/2 reloads them from their original .EXE or .DLL files. (An exception to this is programs that were executed from removable media. OS/2 preloads all pure segments from such .EXE and .DLL files and swaps them as necesary. This prevents certain deadlock problems involving the hard error daemon and the volume management code.)

Because not all segments of a process need to be in memory for the process to execute, we don't have to ensure enough free RAM for the entire process, just enough so that we can simultaneously load six 64 KB segments—the maximum amount of memory needed to run any process. The numbers 6 and 64 KB are derived from the design of the 80286. To execute even a single instruction of a process, all the segments selected by the four segment registers must be in memory. The other two necessary segments come from the worst case scenario of a program trying to execute a far return instruction from a ring 2 segment. The four segments named in the registers must be present for the instruction to start, and the two new segments—CS and SS—that the far return instruction will reference must be present for the instruction to complete. That makes six; the 64 KB comes from the maximum size a segment can reach. As a result, as long as OS/2 can free up those six 64 KB memory regions, by swapping and discarding if necessary, any swapped-out program can execute.

Naturally, if that were the only available memory and it had to be shared by all running processes, system response would be very poor. Normally, much more RAM space is available. The memory overcommit code is concerned only that all processes *can* run; it won't refuse to start a process because it might execute slowly. It could be that the applications that a particular user runs and the usage pattern are such that the user finds the performance acceptable and thus hasn't bought more memory. Or perhaps the slowness is a rare occurrence, and the user is willing to accept it just this once. In general, if the system thrashes—spends too much time swapping—it's a soft failure: The user knows what's wrong, the user knows what to do to make it get better (run fewer programs or buy more memory), and the user can meanwhile continue to work.

Clearly, because all segments of the applications are swappable and because we've ensured that the swap space is sufficient for all of them, initiating a new process doesn't consume any of our free or freeable RAM. It's the device drivers and their ability to allocate nonswappable segments that can drain the RAM pool. For this reason, OS/2 may refuse to load a device driver or to honor a device driver's memory allocation request if to do so would leave fewer than six 64 KB areas of RAM available.

Swapping Miscellany

The system swap space consists of a special file, called SWAPPER.DAT, created at boot time. The location of the file is described in the CONFIG.SYS file. OS/2 may not allocate the entire maximum size of the swap file initially; instead, it may allocate a smaller size and grow the swap file to its maximum size if needed. The swap file may grow, but in OS/2 version 1.0 it never shrinks.

The available swap space in the system is more than the maximum size of the swap file; it also includes extra RAM. Clearly, a system with 8 MB of RAM and a 200 KB swap file should be able to run programs that consume more than 200 KB. After the memory consumed by nonswappable segments and our six 64 KB reserved areas are set aside, the remaining RAM is considered part of the swap file for memory overcommit accounting purposes.

We mentioned in passing that memory used in real mode can't be swapped. This means that the entire 3x box memory area is nonswappable. In fact, the casual attitude of MS-DOS applications toward memory allocation forces OS/2 to keep a strict boundary between real mode and protect mode memory. Memory below the RMSIZE value specified in CONFIG.SYS belongs exclusively to the real mode program, minus that consumed by the device drivers and the parts of the OS/2 kernel that run in real mode.

Early in the development of OS/2, attempts were made to put protect mode segments into any unused real mode memory, but we abandoned this approach. First, because the risk was great that the real mode program might overwrite part of the segment. Although this is technically a bug on the part of the real mode application, such bugs generally do not affect program execution in an MS-DOS environment because that memory is unused at the time. Thus, such bugs un-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

doubtedly exist unnoticed in today's MS-DOS applications, waiting to wreak havoc in the OS/2 environment.

A second reason concerns existing real mode applications having been written for a single-tasking environment. Such an application commonly asks for 1 MB of memory, a request that must be refused. The refusal, however, also specifies the amount of memory available at the time of the call. Real mode applications then turn around and ask for *that* amount, but they don't check to see if an "insufficient memory" error code was returned from the second call. After all, how could such a code be returned? The operating system has just said that the memory was available. This coding sequence can cause disaster in a multitasking environment where the memory might have been allocated elsewhere between the first and second call from the application. This is another reason OS/2 sets aside a fixed region of memory for the 3x box and never uses it for other purposes, even if it appears to be idle.

We mentioned that OS/2's primary concern is that programs be able to execute at all; whether they execute *well* is the user's problem. This approach is acceptable because OS/2 is a single-user system. Multiuser systems need to deal with thrashing situations because the users that suffer from thrashing may not be the ones who created it and may be powerless to alleviate it. In a single-user environment, however, the user is responsible for the load that caused the thrashing, the user is the one who is suffering from it, and the user is the one who can fix the situation by buying more RAM or terminating a few applications. Nevertheless, applications with considerable memory needs should be written so as to minimize their impact on the system swapper.

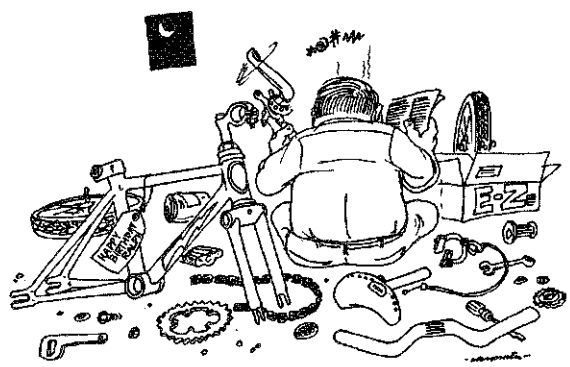
Fundamentally, all swapping optimization techniques boil down to one issue: *locality of reference*. This means keeping the memory locations that are referenced near one another in time and in space. If your program supports five functions, put the code of each function in a separate segment, with another segment holding common code. The user can then work with one function, and the other segments can be swapped. If each function had some code in each of five segments, all segments would have to be in memory at all times.

A large body of literature deals with these issues because of the prevalence of virtual memory systems in the mainframe environment. Most of this work was done when RAM was very expensive. To precisely determine which segments or pages should be resident and which should be swapped was worth a great deal of effort. Memory was costly, and swapping devices were fast, so algorithms were designed to "crank the screws down tight" and free up as much memory as possible. After all, if they misjudged and swapped something that was needed soon, it could be brought back in quickly. The OS/2 environment is inverted: RAM is comparatively cheap, and the swapping disk, being the regular system hard disk, is comparatively slow. Consequently, OS/2's swapping strategy is to identify segments that are clearly idle and swap them (because cheap RAM doesn't mean *free* RAM) but not to judge things so closely that segments are frequently swapped when they should not be.

A key concept derived from this classic virtual memory work is that of the *working set*. A thread's working set is the set of segments it will reference "soon"—in the next several seconds or few minutes. Programmers should analyze their code to determine its working sets; obviously the set of segments in the working set will vary with the work the application is doing. Code and data should be arranged between segments so that the size of each common working set consists of a minimum amount of memory. For example, if a program contains extensive code and data to deal with uncommon error situations, these items should reside in separate segments so that they aren't resident except when needed. You don't want to burden the system with too many segments; two functions that are frequently used together should occupy the same segment, but large unrelated bodies of code and data should have their own segments or be grouped with other items that are in their working set. Consider segment size when packing items into segments. Too many small segments increase system overhead; large segments decrease the efficiency of the swap mechanism. Splitting a segment in two doesn't make sense if all code in the segment belongs to the same working set, but it does make sense to split large bodies of unrelated code and data.

Programmers writing memory-intensive applications should study the literature and their programs to optimize their performance in an OS/2 environment. Minimizing an application's memory requirements is more than being a "good citizen"; the smaller a program's working set, the better it will run when the system load picks up. ♦

Inside OS/2
by Gordon Letwin
Microsoft Press
16011 N.E. 36th Way
Box 97017
Redmond, WA 98073-9717
(800) 638-3030
$19.95



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Virtualizing the 386

by
Kevin Smith

Both the 80386 and the 68020 architectures offer the possibility of developing a virtual machine *hypervisor* that would allow multiple operating systems to run on the same hardware concurrently. At first glance, it would appear that the 68020/30 chip design is better suited to this idea, but the fact that the 80386 machines support a variety of operating systems makes this idea attractive for that chip, too. Each operating system would believe that it had the hardware to itself when in fact each operating system would share real machine resources (memory, I/O ports, CPU time, and privileged instructions) under the direction of a control program. The hypervisor could run as a standalone program or be a component of another operating system (such as Unix).

Ideally, it should be possible to boot the hypervisor under itself. The second (nonprivileged) hypervisor would be unaware that it was running as a "guest" and could itself support yet another hypervisor. Booting a hypervisor under itself would be a "stress" test of the virtual machine software/hardware implementation.

A scenario such as this would, of course, have performance implications, but it would allow for debugging, testing, and evaluation of any operating system (OS) capable of running on the 80386 without having dedicated hardware or "standalone" time. It would also allow users who have OS sensitive applications to run those applications under their required OS while concurrently running new (or just different) OS's and applications. An example of this would be the simultaneous execution of PC-DOS 3.2, PC-DOS 1.1, Xenix-286, Xenix-386, and Unix-386—all running in a nonprivileged state under control of a hypervisor. It would also be possible to obtain multitasking by running multiple copies of a single-tasking OS (such as PC-DOS).

With a hypervisor, old OS's never die, and users aren't forced into difficult conversions just because of new hardware.

A system like this is available for 370 mainframes from IBM (VM/370). It's been around for some 14 years and has been enormously popular. The main attractions of VM/370 have been the ability to test potentially buggy new OS releases in a production environment and to run older OS's that have applications where conversion isn't practical. Large scale multitasking is achieved by running multiple copies of OS's that are otherwise limited in this regard. In effect, it lets users take advantage of new hardware independent of OS restrictions.

The 386 offers the potential for a VM/370-like system. For 8086 applications and OS's, the 386 in its current form should totally isolate the guest OS from hardware reality (with decent performance). Unfortunately, running a 286 or 386 OS as a "guest" presents problems with the current version of the 386. It's essential that a guest OS not be aware that it's running under a hypervisor (unless the hypervisor specifically authorizes this). This requirement is met for virtual-8086 OS's, but appears not to be met for a guest protected mode OS (286 or 386).

This presents a particular problem with operating systems such as OS/2 (80286) and the presumed release of OS/3 (80386). The ability to simultaneously execute these OS's (and their applications) concurrent with Unix will not be possible. The only approach a Unix supplier could take would be to recreate all the new OS/2 and OS/3 system calls in the Unix kernel.

And if any new calls are added in subsequent releases, the Unix kernel will have to be updated again. If the 80286 and 80386 can be virtualized in the same way as the virtual 8086 mode, emulation problems, bugs, etc., can be prevented since the entire 80286/80386 OS will be run as a task under (or with) Unix.

Difficulties

Some of the specific difficulties in "virtualizing" the 80286 and 80386 are

No way to "virtualize" the system flags. PUSHF, as well as flags stored during an interrupt, returns the real status of the Interrupt Flag (IF) and the I/O Privilege Level (IOPL). Because of this, the guest OS is able to determine its true nonprivileged state (which can lead to a guest system's "hanging" due to confusion during a validity check procedure). There is no way to pass the guest "virtualized" system flags. POPF will not allow IF, IOPL, or VM to be modified unless there is sufficient privilege; however, no exception is produced if any attempt is made to alter the system flags (through POPF or IRET). If the guest, believing itself to be the controlling OS, attempts to modify the system flags, the lack of an exception being generated by hardware prevents the hypervisor from determining the current "virtual IF" state of the guest. It would be possible for the hypervisor to pass an interrupt to the guest OS when it has placed itself in what it thinks is a disabled state. This could lead to disastrous consequences if the hypervisor passes an interrupt to the guest at an inopportune time.

Conflicting linear addresses. The guest OS may attempt to use linear addresses that the host is also using. There doesn't appear to be any method of preventing this type of conflict. When physical addresses are used to reference hardware dependent control blocks, the technique of "shadow paging" can

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

prevent this type of problem; however, shadow techniques don't work with virtual (or linear) addresses, which the 386 uses. For example, both the host and guest could be attempting to place a TSS at linear location 8000000. There is no straightforward way, short of actually modifying one of the OS's, to resolve this conflict.

Unpredictable guest/host conflict with the Global Descriptor Table and the Interrupt Descriptor Table (the GDT and the IDT). Both the guest and the host may be attempting to use identical entries in the GDT or IDT for different purposes. For example, both the guest and the host use GDT selector number 30. The host uses it as a descriptor for a paging routine, the guest uses it as a task gate. As with linear addresses, there is no way for the host OS to resolve this.

System hangs caused by nonprivileged system register store instructions. Store instructions for the TR, GDT, IDT, and MSW are not privileged. This allows the guest OS to get the address of a control structure being used by the host OS. The value the guest receives may not bear any resemblance to what it expects, which could lead to a guest system hang. The hypervisor has no way to intercept these types of instructions and return "virtualized" values.

Slow performance due to lack of hardware assist for OS controlled linear to physical address translation. For tasks such as I/O bound jobs, where lots of software directed linear to real mapping is required, the throughput could be unacceptably slow. This is a significant performance issue. It's possible to handle this activity through system software; however, it's very slow because hundreds of instructions are required for each translation. This type of activity can occur many times a second. A new instruction for rapidly translating a linear address to a physical address would be a tremendous performance enhancer. This instruction is described in the suggested change approach later in this article. With hardware help, the translation overhead can be handled with a single instruction.

A High Performance VM Implementation

Two new modes of operation should be introduced. We'll designate the first as "protected normal" mode—identical to the current 80386 "native" mode. The second mode we'll call "protected VM" mode—very similar to native mode except that certain applications are trapped and a "hidden" TSS is used to control this mode that is addressed directly through physical real addresses.

A new type of TSS should be introduced. The TSS selector should be type A (currently reserved), which would be known as the Extended TSS (or XTSS). Bytes 0–67 of the XTSS would match the current TSS. Starting at offset 68 in the XTSS, new fields would be introduced:

68 6 byte savearea for GDTR
6E 2 byte reserved
70 6 byte savearea for IDTR
76 2 byte savearea for TR
78 4 byte CR0 value to be loaded for incoming TSS when "mode switch" occurs
7C 4 byte savearea for DR7 value

Fields used only by the XTSS for the "protected VM"mode:

80 2 byte extended errorcode
82 2 byte reserved
84 4 byte physical address of "protected normal" mode page directory table
88 8 byte TR value (complete TR, including "shadow" information, selector number, base, limit, attributes) for "protected normal" mode XTSS
90 4 byte "virtual EFLAGS"
94 4 byte "virtual CR3" value
98 4 byte linear fault address

The XTSS is used for the two modes "protected normal" and "protected VM." Any programs or OS's that depend on the current TSS structure are not impacted by the XTSS. Only OS's that take advantage of the "protected normal" to "protected VM" transition need to be aware of the XTSS.

A TSS descriptor for the "protected VM" mode XTSS appears in the GDT of the "protected normal" mode operating system. A TSS descriptor for the "protected normal" mode XTSS does *not* appear in the GDT of the "protected VM" mode operating system. The "protected normal"mode is the host, and the "protected VM" mode is the guest. The guest is presumed to be any "off the shelf" 8086, 80286, or 80386 operating system, including the VM operating system itself. It should always be assumed that the guest doesn't realize its true "protected VM" status.

Addressability to the "protected VM" mode XTSS is accomplished by making the following requirement: The contents of CR3 minus 8 is the physical address of a TSS descriptor for the "protected VM" XTSS. If the mode of the machine is "protected VM," this condition is always true—it's the responsibility of the host OS to make sure that the expectation of this condition by hardware is satisfied. Furthermore, the segment base address of this special XTSS descriptor is physical, not linear. This special arrangement allows the guest OS to have linear addresses that are completely independent of any host OS considerations.

If space for a new control register is available, it would be preferable to have a control register (for example, CR1) point to the XTSS; however, it's been assumed that register stack space is at a premium, which is the reason that the CR3 prefix approach is recommended.

The CR3 value (at offset 1C) in the "protected VM" mode XTSS is the address of a page directory table that has the "protected VM" XTSS descriptor 8 bytes in front of it. It's the responsibility of the hosting OS to guarantee that the "protected VM" XTSS is contained in physically contiguous pages if the XTSS crosses a page boundary. It can be assumed

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

that the "protected VM" XTSS CR3 points to a page table structure that has no linear addresses in common with the "protected normal" mode XTSS.

Transition from "protected normal" mode to "protected VM" mode is accomplished by a long JMP to the "protected VM" XTSS descriptor contained in the host's GDT (in "protected normal" mode, the addresses in this descriptor are linear, not physical). The sequence of events for transition to "protected VM" mode are outlined in Figure 1.

CPL must be 0, "protected normal" mode (else general protection exception).
Save all regs in current TSS (as usual); EIP is set pointing to the instruction that *follows* the JMP.
Save GDTR, IDTR, CR0, and DR7 in "protected normal" XTSS (the current TSS plus new fields).
Load new CR3 value from "protected VM" XTSS.
Disable paging bit in CR0.
Get physical addressability to "protected VM" XTSS by using descriptor at CR3 minus 8.
Save complete TR information (including "shadow information" at offset 88).
Load all usual regs from "protected VM" XTSS.
Load GDTR, IDTR, and TR from "protected VM" XTSS. TR value applies to guest OS TSS, not to "protected VM" XTSS.
Load DR7 from "protected VM" XTSS.
Load CR0 from "protected VM" XTSS; force "protected VM" bit and paging enable bit on.
JMP to CS:EIP (or CS:IP) according to information found in "new" GDT/LDT.

Figure 1. Transition to "protected VM" mode.

Two classes of events can trigger a change from "protected VM" mode to "protected normal" mode. The first class are external events. These include page faults, NMI, INTR, or coprocessor related interrupts. The second class of mode switch involves "protected VM" exceptions. These would include such operations as storing/fetching a control register, loading a new MSW (for 286 mode), etc. Transition from "protected VM" mode to "protected real" mode for external events is outlined in Figure 2 on the following page.

Several new bits need to be introduced in CR0:

Bit 30, PVM ("protected VM") bit. When this bit is one, the machine is in "protected VM" mode and is subject to additional exceptions and checks as well as to running under control of the "hidden" extended TSS (addressed by a descriptor containing physical addresses at CR3 minus 8). The IOPL value in EFLAGS is ignored when this bit is on as far as control of I/O ports of the IF flag is concerned. Control of I/O ports is accomplished through an I/O permission bitmap addressed by the XTSS. IOPL is used by hardware for the guest OS to make CPL/IOPL integrity checks; however, when the guest = IOPL, an operation that would otherwise be permitted in "protected normal" mode will be handled in a different fashion (described below).

Bit 29, CI (check IF bit). If PVM = 1 and CI = 0, the current program will not be allowed to alter the IF flag. If the current program makes a transition from enabled to disabled, the "virtual EFLAGS" in the XTSS will be updated, but the "real" IF

DISK FORMAT CONVERSION

XenoCopy-PC

$79.95 + $5.00 S/H
Sales Tax if CA.

PC-DOS program lets your PC

VISA MasterCard UPS COD

READ / WRITE /
FORMAT / DUPLICATE
Disks from over 300 other micros

Upgrades available from all previous versions
for only $25.00 Call for Authorization

To Order Contact:
XENOSOFT™
2210 Sixth Street, Berkeley, CA 94710
(415) 644-9366

Number 123 on the Reader Service Card

Hercules™
★ InColor Card ★
★ Graphics Card Plus ★
★ Graphics Card ★

Portable C and Assembler Drivers to exploit the many virtues of the **Hercules™ InColor™ Card** (GB222), **Graphics Card Plus** (GB112) and **Graphics Card** (GB102).

Routines include Functions and Macros to implement **Text**, **RamFont™** and **Graphics** Modes in **Color** and **Monochrome**. All Registers are maintained mnemonically, and comprehensive C and Assembler "Include" Files are also provided. All are compiler and assembler independent in the Intel® 80x86 environment.

An **Emulation Library** extending Microsoft's QuickC™ / C5.0 Graphics Library to the Hercules™ Cards. The Microsoft® compiler(s) are required to use this library.

Soft Evolutions, Inc.
484 Lake Park Ave., #17
Oakland, CA 94610-2730
(415) 523-8247

$34.95 Postpaid

Number 177 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

KEEP UP WITH THE OS/'s

Megabytes of Memory and 32-Bit Performance for DOS.

If you thought the only way to protected mode was the big move to OS/2...We have good news! You can gain the benefits of protected mode the easy way with OS/286™ and OS/386™. These tools for C, Fortran, Pascal and assembly language programmers permit rapid conversion of existing DOS applications from "real" 8086 mode to "protected" 286 and 386 mode. They don't replace or modify DOS, but extend it to protected mode. This way you get multi-megabytes of directly addressable memory (16Mb-286, 4Gb-386) with the compiler, TSRs, device drivers, graphic routines, etc. you use today.

OS/286 and OS/386 are the only DOS extenders that span both the 286 and 386 processors. They run on the widest array of AT and 386 machines, with 32-bit capability *today* on 386s that yields twice the performance of 16-bit mode.

OS/286 and OS/386 are the preferred solutions for developers of high performance memory-intensive applications, including CADKEY, CASE, and Gold Hill, and premier language developers Lahey and MetaWare.

Our optional TOUCHDOWN™ BIOS supplement provides fast and reliable protected mode operation on *any* 286 system, even those with problems resetting the 286. (Ever notice how few existing machines Operating System/2 runs on?) TOUCHDOWN is not required for most major brand AT clones, but for the older machines it is a lifesaver!

If your applications are running out of memory or need more speed, enhance them now without abandoning your investment in DOS.

Coming Soon
WINDOWS 2.0
and DESQview 2.0 API support

Give your protected mode DOS a powerful presentation layer like OS/2, SAA or DESQview.

Special $49.50 Evaluation Offer

Check out for yourself the benefits of protected mode. Our $49.50 "sampler special" is a complete OS/286 Developer's Kit, but with a time limited, non-distributable kernel. There's no better way to learn about the outstanding features of OS/286 and OS/386 than to try them. Of course, the $49.50 is applicable to the purchase of the full OS/286 (16-bit) or OS/386 (32-bit) kit at $495 for either one.

The OS/x86 Developer's Kits include support for popular, C, Fortran, Pascal, Lisp, Prolog, compilers, and assemblers from: MetaWare, Lahey, Microsoft, Lattice, Gold Hill, LogicWare and Phar Lap. A number of other packages are also supported including PLINK86, Halo & GSS Graphics, DESQview and soon Windows 2.0.

Run time licenses for OS/286 and OS/386 are available from $40/copy to under $1.00.

The HummingBoard® turns any XT, AT or 386 into the fastest, most capable 386 system available. The dual processor architecture provides up to 900K each for multiple real mode applications (Lotus, etc.) which can co-reside with protected mode applications on the HummingBoard's 20MHz 386, while debuggers, editors, networks run concurrently on the base processor. 1-24MB memory, 8,870 Dhrystones. A MUST TRY!

A.I. Architects, Inc.

One Kendall Square, Cambridge, MA 02139
TEL (617) 577-8052 FAX (617) 577-9774
Credit card orders only – 24 hours. 617-577-1305

HummingBoard® is a registered trademark and OS/286, OS/386 and TOUCHDOWN are trademarks of A.I. Architects, Inc., PLINK86 is a trademark of Phoenix Corp., HALO is a registered trademark of Media Cybernetics, Inc., DESQview is a trademark of Quarterdeck Office Systems, Windows 2.0 is a trademark of Microsoft Corp.

Number 180 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Force the paging enable bit in CR0 off.
Accomplish addressability to the "protected VM" XTSS by fetching the physical address of the XTSS from the descriptor found at CR3 minus 8.
Save all current regs in the XTSS.
Save the GDTR, IDTR, TR, and CR0.
Save DR7.
Set the extended errorcode (offset 80) to 0.
Load TR with the complete value from offset 88 (this includes "shadow information")—this is the "protected normal" XTSS value.
Load CR3 with the "protected normal" page directory value at offset 84.
Force the paging enable bit in CR0 on, the "protected VM" bit off.
Load the GDTR, IDTR, and CR0 from the "protected normal" XTSS.
Load DR7.
Load other XTSS registers.
If the interrupt is through a task gate, set the TR backlink field to the current value (the selector number for the "protected normal" XTSS).
CS:EIP in the "protected normal" XTSS points to the instruction following the JMP that caused the transition to "protected VM" mode. It's the responsibility of the program that caused the transition to be aware of the implications of this.
Process interrupt through IDT in normal fashion.

Figure 2. Transition from "protected VM" mode to "protected real" mode for external events.

flag will not be altered. The "virtual EFLAGS" will match the "real EFLAGS" except that the IF flag in the "virtual EFLAGS" field will be altered and the real IF flag won't be altered. This applies to an enabled to disabled transition caused by CLI, POPF, IRET, a task switch, or going through an interrupt gate. The hypervisor will determine if the guest OS thinks it disabled by examining the "virtual EFLAGS" in the XTSS. As long as this interrogation is done by the hypervisor, no context switch to the hypervisor is necessary for the guest enabled to disabled transition. This is a performance consideration. When a guest disabled to enabled transition occurs and PVM = 1 with CI = 0, interrupt 17 (currently reserved) should be issued.

The extended error code in the XTSS should be set to 0001 with the linear fault address in the XTSS set to the address of the instruction that caused the transition. CS:EIP (or CS:IP if 16 bit/80286 mode) in the XTSS should point to the instruction that *follows* the one causing the switch to enabled mode. Causing a context switch to the hypervisor for the guest disabled to enabled transition is necessary since it's possible that the hypervisor has been queueing incoming interrupts intended for the guest while it's been in its "virtual disabled" state. The IF bit in the "virtual EFLAGS" should also be updated by hardware. Also, the PVM = 1 and CI = 0 state should cause the guest OS to receive the "virtual EFLAGS"—not the "real" EFLAGS—as the result of a PUSHF, interrupt, or task transition.

If PVM = 1 and CI = 1, the guest OS is granted control of IF in the "real" EFLAGS register. The "virtual EFLAGS" continue to be maintained regardless of the CI setting.

Software Developers

We need your program!

Do you have a program that's good enough to sell, but don't want the problems or financial risk of producing, typesetting, printing, packaging, warehousing, marketing, distributing and supporting a product?

Why start your own software house?
We've done it for you!

Merlin Publishing Group
is now accepting submissions of micro-computer software for publication.

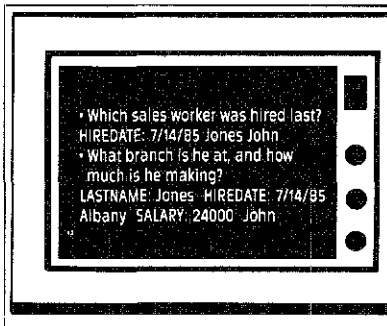
You get:
- initial cash payments
- generous royalties
- to spend your time programming

But we can't help you if you don't submit. Call or write today for our submission guideline kit.

MERLIN
PUBLISHING GROUP ™
1240 Johnsons Ferry Place, Suite A10
Marietta, GA 30068
(404) 977-6034

P. S. See us in Atlanta at COMDEX-Spring '88

Number **141** on the Reader Service Card


• Which sales worker was hired last?
HIREDATE: 7/14/85 Jones John
• What branch is he at, and how much is he making?
LASTNAME: Jones HIREDATE: 7/14/85
Albany SALARY: 24000 John

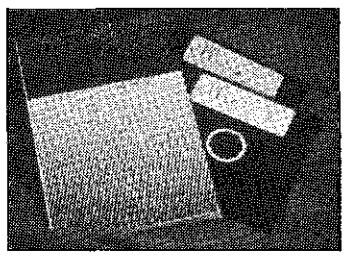
announcing
ellie
version 2.0

English in, dBASE out

. . . save hours of coding!
Try the amazing English-language interface for dBASE III/III+. (Now Fox Base + compatible!) 30-day money-back offer. Satisfaction guaranteed. Call 212-316-9078

"supports queries . . . that would otherwise require extensive JOIN and SET RELATION coding in dBASE." —*PCAI*, 1/88

"Using *ellie*, you can almost forget about dBASE commands." —*PCM*, February '88



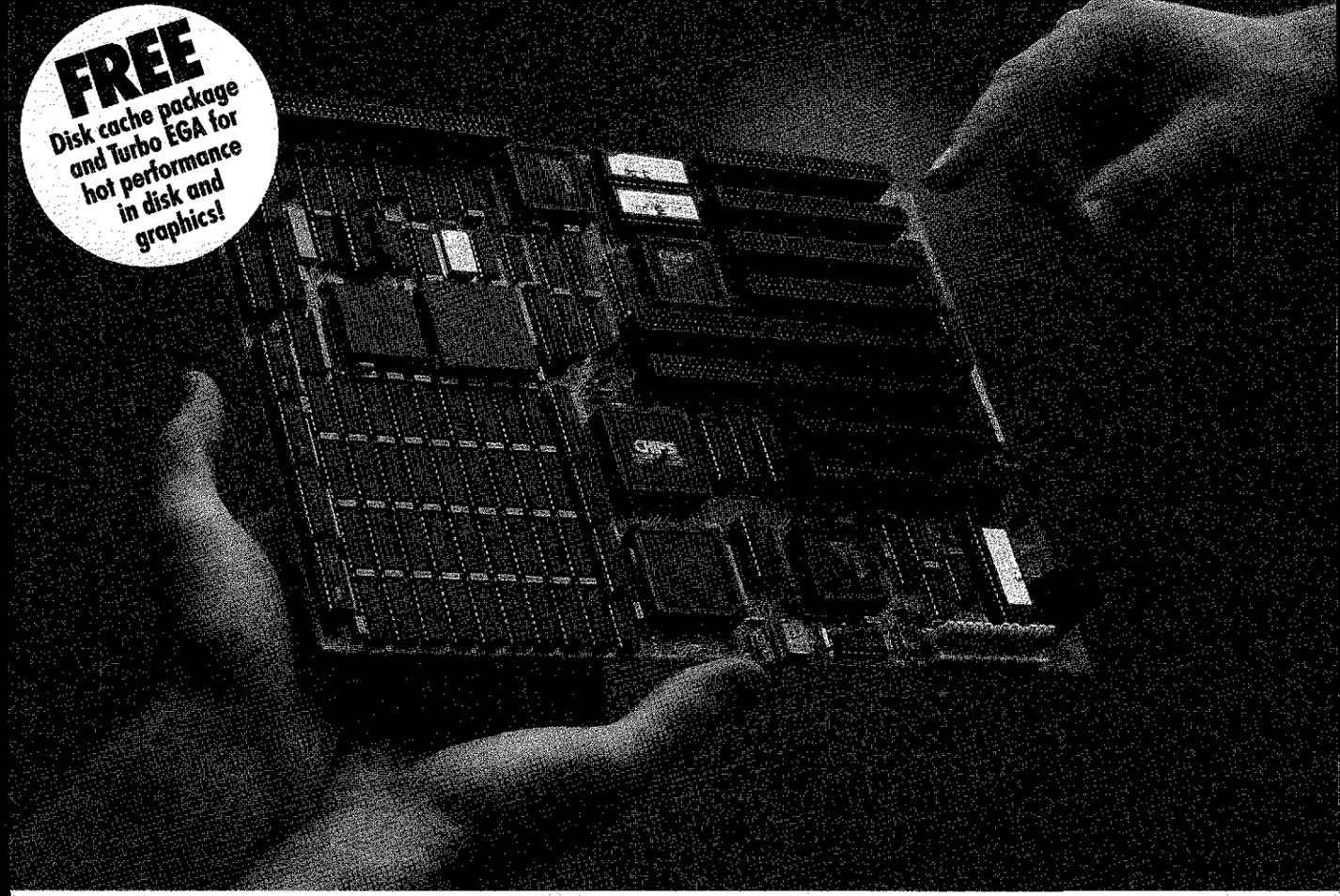
only $99

ELFSOFT
210 W 101 NY NY 10025

Dealer inquiries invited.

Number **113** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



CONVERT YOUR PC, XT OR AT INTO A HIGHER FORM OF LIFE!

386 MOTHERBOARDS FOR 386 SPEED

Don't let your PC give up the ghost—Hauppauge has just arrived with a new spark of life: the 386 MotherBoard.™ Far more advanced than an accelerator card, our line of MotherBoards grace your PC, PC/XT, PC/AT or compatible with speeds equal to the IBM PS2 Model 80. And faster. Because we've built in 1 Megabyte of high speed RAM and a 387 math coprocessor socket for speeds that make users humble with awe.

OS/2 Compatible. To ensure a long, fruitful life, our 386 MotherBoard is compatible with the PC/AT (BIOS and I/O)—so you can run the new generation of DOS, OS/2. Our Board also runs Windows/386, UNIX V and PC-MOS/386. For more power, you'll find 16-bit expansion slots that accommodate the latest I/O expansion cards. No 386 accelerator card gives you so much versatility. **Only our 386 MotherBoard gives your PC a future with unlimited possibilities!**

The Critics Applaude! *PC Magazine* awarded our Board "The Editor's Choice" for 386 Replacement Boards. *PC World* called it "the Upgrade Product of the Year."

Technical Features ▪ 16 MHZ 80386 ▪ 1 Megabyte of 100 nsec 4-way interleaved RAM ▪ PC/AT compatible I/O and BIOS for support of OS/2 ▪ Six 8-bit expansion slots ▪ two 16-bit expansion slots (four on 386 MotherBoard/AT) ▪ One 32-bit expansion slot for up to 12 Megabytes of high speed memory ▪ Battery-powered clock calendar

386 MotherBoard/PC or MotherBoard/XT	$1495
386 MotherBoard/AT	$1595
32-bit RAM Board (2 Mbytes installed; up to 4 Mbytes)	$795
16 MHz 80387 math coprocessor	$695
16-bit combination hard disk/ floppy disk controller	$245

For more information on our easy-to-install MotherBoard, call: **1 (800) 443-6284**. In New York, call **(516) 434-1600**.

Hauppauge Computer Works, Inc.
175 Commerce Drive,
Hauppauge, New York 11788

Hauppauge!

Number 122 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

In all cases, the EFLAGS value stored at offset 24 in the XTSS should be the real "EFLAGS" value. The "virtual EFLAGS" are used only when the guest OS is storing/fetching EFLAGS. The EFLAGS value stored in a guest OS TSS would be the "virtual EFLAGS."

Bit 28, TI (trap on interrupt bit). If PVM = 1 and TI = 1, any software interrupt (INT n) issued by the guest, or issued by hardware because of an error condition (for example, interrupt 6, invalid opcode), will cause a transition to the hypervisor via INT 17. The low order byte of the extend errorcode in the XTSS should be set to 02 with the high order byte set to the interrupt number that triggered the transition from "protected VM" mode to "protected normal" mode. The linear fault address in the XTSS should point to the instruction that caused the transition. The CS:EIP (or CS:IP) value in the XTSS should point to the first instruction to be executed after the interrupt is taken. This facility allows the hypervisor to monitor all interrupts taken by the guest. Under most conditions, this wouldn't be desirable due to performance implications.

Transition from "protected VM" mode to "protected real" mode due to a "protected VM" privilege exception is outlined in Figure 3.

Force the paging enable bit in CR0 off.
Accomplish addressability to the "protected VM" XTSS by fetching the physical address of the XTSS from the descriptor found at CR3 minus 8.
Save all current regs in the XTSS.
Save the GDTR, IDTR, TR, and CR0.
Save DR7.
Set the extended errorcode (offset 80) to the appropriate value (described below).
Update the linear fault address (offset 98).
Load TR with the complete value from offset 88 (this includes "shadow information"—the "protected normal" XTSS value.
Load CR3 with the "protected normal" page directory value at offset 84.
Force the paging enable bit in CR0 on; force the "protected VM" bit off.
Load the GDTR, IDTR, and CR0 from the "protected normal" XTSS.
Load DR7.
Load other XTSS registers.
If the interrupt is through a task gate, set the TR backlink field to the current value (the selector number for the "protected normal" XTSS points to the instruction following the JMP that caused the transition to "protected VM" mode. It's the responsibility of the program that caused the transition to be aware of the implications of this.
Issue interrupt 17.

Figure 3. Transition from "protected VM" mode to "protected real" mode due to a "protected VM" privilege exception.

The following instructions and conditions are affected if the PVM bit in CR0 is 1. Any exceptions caused by PVM's being 1 result in a switch back to "protected normal" mode. INT 17 (currently reserved) will then be issued once the switch to "protected normal" mode is made. Any INT 17 exception will cause an extended error code to be set in the "protected VM" mode XTSS. All CPL/IOPL checks that apply to "protected normal" mode also apply to "protected VM" mode; however, in "protected VM" mode, an instruction or state change that would have otherwise occurred may be trapped. The instructions and conditions affected by PVM appear in Figure 4.

LMSW	Extended errorcode 3
SMSW	Extended errorcode 4
MOV CRn,reg	Extended errorcode 5
MOV reg,CRn	Extended errorcode 6
MOV DRn,reg	Extended errorcode 7
MOV reg,DRn	Extended errorcode 8
MOV TRn,reg	Extended errorcode 9
MOV reg,TRn	Extended errorcode 10
CLTS	Extended errorcode 11
HLT	Extended errorcode 12
TRA (new)	Extended errorcode 13
IN/OUT	I/O permission bitmap

Figure 4. The instructions and conditions affected by PVM.

All of the above instructions cause the CS:EIP in the "protected VM" XTSS to point to the instruction that follows the one causing the exception (trap style). The linear fault address points to the instruction that caused the exception.

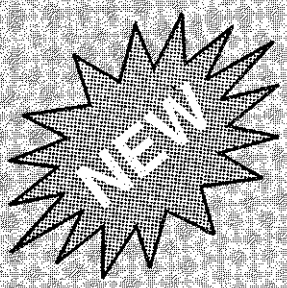
IN, INS, OUT, and OUTS instructions are affected by the I/O permission bitmap defined in the "protected VM" XTSS. If the guest OS is executing under a TSS that also defines an I/O permission bitmap, the corresponding bits for both maps should be OR'ed together. If the result of the OR is 1, INT 17 with an error code of 20 should be issued. CS:EIP in the "protected VM" XTSS should point to the instruction that follows the one that caused the exception. If the guest TSS doesn't define an I/O permission bitmap, or if the guest TSS is 80286 style, the input to the OR operation should be treated as 0 (only the XTSS I/O permission bit is relevant for this case).

PUSHF, POPF, IRET, CLI, and STI should store/fetch the "virtual EFLAGS" value stored at offset 90 in the XTSS. The "virtual EFLAGS" matches the "real" EFLAGS with the exception of the IF (interrupt flag) bit. Unless CI is on (bit 29 in CR0), a "protected VM" task is never allowed to inspect/modify the real IF flag. If the "protected VM" task changes from logically enabled to logically disabled, no exception will occur—only the "virtual EFLAGS" will be updated. If the task goes from logically disabled to logically enabled, INT 17 with an extended errorcode of 21 should be issued. CS:EIP in the "protected VM" XTSS should point to the instruction that follows the one that caused the exception. The linear fault address points to the instruction that caused the exception. IF in the "virtual EFLAGS" will not be altered unless the usual protection checks are satisfied (IOPL not less than CPL, etc.). The "virtual EFLAGS" maintenance should also occur for guest task switches as well as for guest interrupts that go through an interrupt gate. The errorcode 21 INT 17 should be issued if either of these events causes the logically disabled to logically enabled transition.

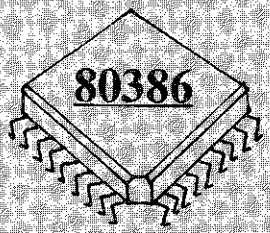
If a task switch occurs and the incoming guest 386 TSS has a CR3 value that differs from the "virtual CR3" in the XTSS (at offset 94), INT 17 with an extended error code of 22 should be issued. CS:EIP in the "protected VM" XTSS should point to the first instruction in the new guest task. The

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SERIOUS DEBUGGING AT A REASONABLE PRICE



Soft-ICE™



All the speed and power of a hardware-assisted debugger at a software price - $386

Features

Real-time break points on:
- Memory locations
- Memory ranges
- Execution
- I/O ports
- Interrupts (hardware and software)

Dual monitor support

System memory map

Regain control with a keystroke
Even with the following code:

```
CLI
MOV   AL,0FFH
OUT   21H,AL
JMP   $
```

and much, much more

How Soft-ICE works

Soft-ICE unleashes the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment. Soft-ICE uses 80386 protected mode features, such as paging, I/O privilege level, and break point registers, to add real-time hardware-level breakpoints to your existing DOS debugger. To use Soft-ICE you simply pop the Soft-ICE window up through a key sequence, set your hard break points, then return to your soft debugger. As the target program is executing, Soft-ICE recognizes when the breakpoint conditions have been reached and gives control back to your soft debugger. And this is all done at full 80386 speed! Soft-ICE can also be used in stand-alone mode. This comes in handy if you are debugging loadable device drivers, interrupt handlers, or terminate and stay resident programs. All of the standard debugging features are available to help you find the most difficult systems problems.

Benefits

- **Works with CodeView** --To get you up and going as fast as possible, Soft-ICE is designed to work with your existing software debuggers such as CodeView and Pericope I & II.
- **Breaks the 640K barrier** -- If you have extended memory, Soft-ICE takes up **ZERO** bytes of memory in the first 1MB of address space. This means you can load and debug your largest programs.
- **Power of an in-circuit emulator** -- At only $386, you can give every member of your software development team the power of an in-circuit emulator.
- **AT compatible 80386 PC's** -- Soft-ICE works with all AT compatible 80386 PC's, such as COMPAQ's Deskpro 386 and the IBM Model 80.
- **Easy to learn** -- Soft-ICE is so easy to learn that you can be finding bugs with Soft-ICE by the time you could install a hardware-assisted debugger.

NEW **New product - MagicCV** Run CodeView in less than 8K . **MagicCV** places CodeView in a second virtual machine. This means that you have over 500K available for your program. -- CALL TODAY

30 day money-back satisfaction guarantee. Visa and Master Card accepted. Ask about our coupon program.

To order or to get more information, call (603) 888-2386

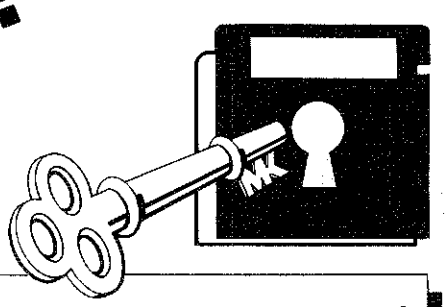
NU-MEGA TECHNOLOGIES

P.O. BOX 7607
NASHUA, NH 03060-7607

Number 160 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MASTER*KEY Unlocks Everything!



turn this into this!

```
C:>DEBUG PROGRAM.COM
-D100 136
8848:0100  EB 18 49 6E 63 6F 72 72-65 63 74 20 44 4F 53 20  k.Incorrect DOS
8848:0110  76 65 72 73 69 6F 6E 0D-0A 24 50 B4 30 CD 21 86  version..$P40N!.
8848:0120  E0 3D 36 01 72 05 3D 0A-02 76 09 BA 02 01 B4 09  `=6.r.=..v.:..4.
8848:0130  CD 21 CD 20 58 EB 2F                             M!M Xk/
-Q
```

```
H00100: JMP     Short H0011A                        ;00100 EB18        __
;-----------------------------------------------
        DB      "Incorrect DOS version"             ;00102 496E636F727265
        DB      0Dh                                 ;00117
        DB      0Ah                                 ;00118
        DB      "$"                                 ;00119 24          .
;-----------------------------------------------
H0011A: PUSH    AX                                  ;0011A 50          P
        MOV     AH,30h                              ;0011B B430        _0
        INT     21h             ;1-DOS_Ver_Number   ;0011D CD21        _!
        XCHG    AH,AL                               ;0011F 86E0        __
        CMP     AX,0136h                            ;00121 3D3601      =6_
        JB      H0012B          ; . . . . . .       ;00124 7205        r_
        CMP     AX,020Ah                            ;00126 3D0A02      =__
        JBE     H00134          ; . . . . . .       ;00129 7609        v_
H0012B: MOV     DX,0102h                            ;0012B BA0201      ___
        MOV     AH,09h                              ;0012E B409        __
        INT     21h             ;1-Display_String   ;00130 CD21        _!
        INT     20h             ;TERM_normally:20h  ;00132 CD20        _
;-----------------------------------------------
H00134: POP     AX                                  ;00134 58          X
        JMP     Short H00166                        ;00135 EB2F        _/
;-----------------------------------------------


MASTER*KEY XREF - PROGRAM.XRF                                   Page 1

0102h                   :       121     2F5     301     320
020Ah                   :       126
03CBh                   :       12B
1-Display_String        :       130     591     610
1-DOS_Ver_Number        :       11D
H00100                  :       100
H0011A                  :       100     11A
H0012B                  :       124     12B
H00134                  :       129     134
H00166                  :       135
TERM_normally:20h       :       132
```

NOTE: The cross-reference is by memory location within the program file!

NOTE: The output is totally Microsoft MASM-compatible.

(not copy protected)

MASTER*KEY No Other Product Comes Close!

An EXPERT may not know the solution, but always knows where to find it.

MASTER*KEY HELPS ANYONE solve those confusing and frustrating software puzzles more rapidly and easily than any other software available, at any cost! It gives you know-how within hours that may otherwise take years of experience. Create a new program from an old one. DON'T REINVENT THE WHEEL!

MASTER*KEY - Smart!
MASTER*KEY is an intelligent self-documenting MS-DOS reverse assembler. Its sophisticated procedures swiftly race through massive and baffling object code files to effortlessly discover potential trouble spots.

MASTER*KEY - Educational!
YOU DON'T NEED TO KNOW ASSEMBLY LANGUAGE! MASTER*KEY will take any program from your IBM-compatible computer and return fully-documented, easily-understood assembly language source code (Microsoft MASM 5.0 compatible).

MASTER*KEY - Easy To Use!
MASTER*KEY works both automatically from the DOS command line or interactively from menus similar to Lotus Corporation's 1-2-3 or Symphony. No need to remember any new commands or continually refer to a manual. Use it immediately!

Minimum System Requirements:
256K + 8088/8086/80186/80286/80386 PC
MS-DOS or PC-DOS 2.0 +
One 360K DSDD Floppy Drive (IBM PC Format)

MS-DOS is a trademark of Microsoft.
PC-DOS is a trademark of IBM.

MASTER*KEY will guide you step by step to:

1. Help you learn assembly language, if you desire.
2. Discover how any program runs or why it doesn't.
3. Alter or remove unwanted object code from any program.
4. Incorporate routines from compiled programs into other assembly language, Basic, C, or Pascal programs.
5. Make software more compatible with your computer. Be certain a questionable program won't damage your system BEFORE you run it.
6. Modify software to operate with other versions of DOS.
7. Customize your COMMAND.COM or other executable program directly or by reassembling your altered MASTER*KEY source code.

Order Now! Just $79.95

Phone orders accepted on MC or VISA
$82.45 (includes $2.50 shipping)
$87.65 in California (includes tax & shipping) **C.O.D.** orders add $2.00

(714) 596-0070

Dealer/Distributor Inquiries Welcome

Sharpe Systems Corporation
2320 E Street, Dept. 1610, La Verne, CA 91750 714-596-0070

MASTER*KEY should not be confused with any public domain or share ware software that may have a similar name or be a similar product.

Please send **MASTER*KEY!**

Send checks to:
Sharpe Systems Corporation
2320 E Street, Dept. 1610, La Verne, CA 91750

Name _____

Address _____

City _____ State _____ Zip _____

Number **175** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

linear fault address points to the instruction that caused the guest task switch. The use of "virtual CR3" is strictly for performance and prevents an otherwise unnecessary INT 17 if the guest OS isn't attempting to change the page table during a task switch.

Paging

Paging is identical for "protected normal" and "protected VM" modes. Whenever a guest OS attempts to build and use its own page tables, the store/fetches of CR3 are trapped and the hypervisor gathers information used to maintain a "shadow" page table.

Since page faults will be handled by the hypervisor in "protected normal" mode, all guest data structures (such as GDT, IDT, TSS, etc.) can be pageable.

A new instruction would be very useful to expedite the chores required by the hypervisor to maintain the "shadow" page tables. This new instruction would be known as TRA (Translate Real Address). The instruction would need two operands, which could be supplied through "hard coded" registers, say EAX (for the first operand) and EDX (for the second operand). This instruction is legal only at CPL 0 in "protected normal" mode. EAX would contain a virtual address; EDX would contain the physical address of a page table directory (not necessarily the current one). The instruction would determine the physical page frame number with this information and return the result in EAX. If the address was invalid, the carry flag would be set to 1; otherwise, if all goes well, the carry flag would be set to 0.

The hypervisor will have to simulate many privileged instructions. The simulation requires that the address of the instruction triggering the exception be known (for inspection by the hypervisor) as well as the address of the next instruction (for process resumption after the proper simulation). A very useful bit of information to have would be the length of an instruction that triggers a fault or trap. This could be known as the ILC (Instruction Length Code). The ILC could use bits 18–21 in EFLAGS (currently reserved). In the case of a fault, the ILC would reflect the length of the next instruction that was to be executed; in the case of a trap, the ILC would reflect the length of the last instruction executed. For example, assume that CLI was issued with insufficient privilege and causes a GP. The return address points to the CLI, and the ILC would be 1. Assume that software interrupt 33 is issued. The return address follows the INT 21h instruction, and the ILC would be 2.

Handling of virtual 8086 mode should be identical for "protected normal" and "protected VM" modes.

The Gains

The 80386 has the potential to totally isolate hardware dependencies from software. The virtual 8086 mode fulfills this potential for 8086 operating systems. Difficulties in "virtualizing" 80286 and 80386 operating systems exist with the current definition of the 80386; however, compared to what's already been accomplished with the 80386, these difficulties are not overwhelming. A VM/Hypervisor concept has been very successful in the mainframe world. Many of the reasons for this success are applicable to the microcomputer/workstation world. Consideration should be given to extending and/or modifying the 80386 architecture to support guest operating systems for the 80286 and 80386 as well as the 8086.

Space for microcode is always a problem. The 386 is a busy chip, but there may be room left for these proposed changes. These are but possible approaches for a full VM system. A totally different implementation may be more realistic given microcode and/or scheduling restraints. Tradeoffs between practicality and performance must be considered. The value, though, of having some method of achieving a full VM system (for 8086, 286, and 386) can't be denied. ♦

References

Publications describing VM/370 (as a working example of the hypervisor concept) are available from IBM:

GC20-1800 *VM/370 Overview*
GC20-1820 *CP Command Reference*
GC20-1806 *VM Operators Guide*
GC20-1807 *Systems Programmers Guide*
GC20-1821 *Operating Systems in a Virtual Machine*
SY20-0884 *Data Areas and Control Block Logic*

Also see

Deitel, Harvey M. *An Introduction to Operating Systems.*

Deitel's book contains an excellent case study on VM.

386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 •

NOW!

C / Pascal → 386

Paradox 386
Foxbase+ 386
386-MATLAB,/Weitek

... and others ...

These and other protected-mode 32-bit 80386 programs are among the first to take advantage of the full power of the 386. They and practically every **386 protected-mode** MS-DOS program that's shipping were done with MetaWare's compilers.

It's no surprise. The recognized leader, MetaWare introduced the *first* **C** and **Pascal** compilers that generate protected-mode 386 code for running on any 386 MS-DOS machine (e.g., the Compaq 386 or the IBM PS/2-80) over a year ago. **High C** ™ and **Professional Pascal** ™ are well-established and proven.

Smart software developers aren't waiting! Industry leaders such as Borland (ANSA) and Fox use MetaWare's compilers to get dramatic increases in speed and functionality. Don't wait years for Microsoft's 386DOS—your competition will have a big jump on you!

Expand your application to the large 32-bit address space and the full 32-bit registers of the 80386. Go with the long-standing leader. Contact MetaWare for your 80386 software solution today!

(408)429-6382 telex 493-0879

MetaWare™ INCORPORATED

903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060

The Clear Choice for Large Programming Projects —PC Tech Jnl

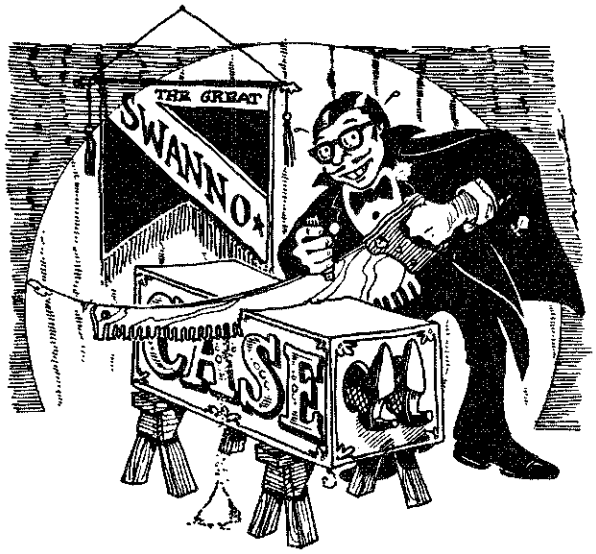
386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 •

Number 45 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

A CASE of the Jumps

by
Tom Swan



Turbo Pascal 4.00 cheats when it compiles a CASE statement. Instead of creating a jump table of CASE-label addresses as you might expect, the compiler generates a series of conditional jumps. This sleight of hand turns a potentially fast CASE into a sluggish series of IF-THEN-ELSE statements—an illusion in which program speed can vanish in a puff of smoke.

To understand the problem, let's saw the lady in half—exposing the innards of a simple Pascal CASE statement. Then, let's look at a similar program in assembly language. Finally, let's reassemble our victim (volunteer) adding fast jump table CASE statements to Turbo Pascal. As you'll see, the result can be a tremendous boost in program speed.

"Hey Rocky, Wanna See Me Pull a Rabbit Out of My Hat? . . . Oops, Wrong Hat!"

Listing 1 at the end of the article illustrates a typical CASE statement in Turbo Pascal. A FOR loop executes the CASE statement four times, selecting among the four labeled statements and calling procedures **proc0** through **proc3**, which display the digits 0–3. Of course, this is a pretty dumb way to display four digits, but we need a volunteer to illustrate the insides of a typical CASE statement and this simple example suits.

Figure 1 disassembles the compiled CASE statement from Listing 1. The first MOV instruction loads AX with the value of the FOR-loop control variable, **i**. After this comes the code for the four labeled Pascal statements. Each of these statements begins by comparing AX with the label's value, 0000, 0001, 0002, or 0003. If the CMP fails, the JNZ instruction jumps to the next comparison or to the end of the CASE statement. If the CMP does not fail, the CALL instruction calls PROC0, PROC1, PROC2, or PROC3—the assembly language labels for the Pascal procedures of the same names. After the CALL, a JMP instruction jumps to the end of the CASE statement.

```
11:           case i of
4202:00C1 A10000          MOV     AX,[0000]
12:                 0 : proc0;
4202:00C4 3D0000          CMP     AX,0000
4202:00C7 7505            JNZ     @+1F (00CE)
4202:00C9 E834FF          CALL    PROC0
4202:00CC EB1C            JMP     @+3B (00EA)
13:                 1 : proc1;
4202:00CE 3D0100          CMP     AX,0001
4202:00D1 7505            JNZ     @+29 (00D8)
4202:00D3 E855FF          CALL    PROC1
4202:00D6 EB12            JMP     @+3B (00EA)
14:                 2 : proc2;
4202:00D8 3D0200          CMP     AX,0002
4202:00DB 7505            JNZ     @+33 (00E2)
4202:00DD E877FF          CALL    PROC2
4202:00E0 EB08            JMP     @+3B (00EA)
15:                 3 : proc3
4202:00E2 3D0300          CMP     AX,0003
4202:00E5 7503            JNZ     @+3B (00EA)
4202:00E7 E899FF          CALL    PROC3
4202:00EA 833E000003      CMP     Word Ptr [0000],+03
4202:00EF 75CC            JNZ     @+0E (00BD)
```

Figure 1: Partial disassembly of Listing 1

From this analysis, even if you're not an assembly language wizard, you can see that it takes four CMP and JNZ instruction pairs to call PROC3, three such instruction pairs to call PROC2, and so on. The CASE statement may seem magically efficient in the Pascal source code, executing labeled statements at random like rabbits pulled from a hat. But it's the wrong hat. Instead of a fast hare, in the compiled code, we get a tedious IF-THEN-ELSE tortoise. And this is only a simple four-label CASE statement. Every additional label adds another CMP and JNZ pair, making the program work harder and harder to reach the last labeled statement.

Improving CASE Statement Speed

Without resorting to assembly language, there are a few prestidigitations you can perform to improve CASE statement speed. One trick, suggested by Jon Greenblatt in "A Note on Optimizing Turbo Pascal" (*PJ*, September/October 1987, pp 42–43), is to place the more frequently used labels nearer to the CASE key word. Of course, the tradeoff is a loss of speed for the labeled statements moved farther down.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Avoiding compound statements is another good idea. For example, consider this CASE statement:

```
CASE i OF
   0 : Proc0;
   1 : BEGIN
          Writeln( 'First line' );
          Writeln( 'Index value = ', i );
          Writeln( 'Index squared = ', i * i );
          Writeln( 'Index cubed = ', i * i * i )
       END;
   2 : Proc2
END;
```

The large compound statement at label 1 prevents Turbo Pascal from using the CMP/JNZ instruction pair as in Figure 1 because JNZ can jump only 127 bytes forward. Instead, the compiler generates the following code to jump to the statement at label 2:

```
CMP AX,0001   ; Is selector = 1?
JZ  @+1D      ; Yes, jump into compound statement
JMP @+C5      ; No, jump to next case label
```

This arrangement requires executing three instructions, CMP, JZ, and JMP, for each label not matching the CASE selector. You can avoid the additional JMP by converting the compound statement into a procedure although this adds the overhead associated with calling subroutines plus the miscellaneous chores performed inside the procedure code itself. Even so, the CASE statement as a whole operates faster with simple labeled statements that take 127 bytes or less.

Beyond these two suggestions—moving frequently used CASE labels as high as possible and avoiding large compound statements—there isn't much you can do to improve a CASE statement's performance. If you're willing to use assembly language, though, with the help of a construction called a jump table, a little machine code is like pixie dust—it can make a CASE statement really fly.

Understanding Jump Tables

If you're queasy about assembly language, the following might give you butterflies. Bear with me. We'll return to Pascal in a moment.

Listing 2 is a MASM 5.0 example of an assembly language jump table—the rough equivalent of the Pascal CASE statement example in Listing 1. The table in the data segment stores the four offset addresses of the labels, **case0**, **case1**, **case2**, and **case3**. After initializing a few items, the program executes the code at one of these labels, jumping indirectly to the appropriate address stored in the table. Save the program as CASE.ASM. Assemble and run the test with these DOS commands:

```
MASM CASE;
LINK CASE;
CASE
```

Type 0, 1, 2, or 3 to execute one of the four labeled cases. Typing anything else ends the program with an error message. Examine the instructions just above **case0**. After getting and testing a keypress, the program loads BL with the digit in AL, then ANDs the result with 03, creating a 16-bit value ranging from 0 to 3. Shifting this value one bit to the left multiplies BX by 2, forming a word index into the address table. After this, the JMP instruction executes the selected case by jumping indirectly to the appropriate address.

Because the same JMP instruction executes all of the labeled sections, **case0** through **case3**, it takes no more time to start running the code at **case3** than it does to start the code at **case0**. Each case is given equal time, a democratic arrangement and primary advantage of jump tables.

Each case is given equal time, a democratic arrangement and primary advantage of jump tables.

Despite this advantage, though, there is one main disadvantage to using jump tables. If your case selector values (the digits in AL in this example) are widely separated, you must insert dummy offsets into the table. For example, suppose your selectors are 0, 2, 4, and 6. The jump table must reserve space for the unused values, 1, 3, and 5. The easiest method is to use the **endcase** label for unused selector values, rewriting the jump table so that the illegal values 1, 3, and 5 simply jump to the end of the case selections:

```
table dw case0, endcase, case1, endcase
      dw case2, endcase, case3
```

With greatly separated values, say only 6 possible values in the range 0 to 255, the number of **endcase** entries for unused values might cause the jump table to grow too large for your requirements. In this event, you must consider whether the time savings of jump tables are worth the wasted memory you can save with a series of conditional jumps.

Writing Jump Tables in Pascal

Okay, we've sawed the lady in two and we've seen what she's got inside. We know how to code a jump table in assembly language. We've seen that jump tables operate more efficiently than compiled Pascal CASE statements by giving equal access to labeled statements. The questions remain: How can we teach Turbo Pascal to use jump tables, and what are the practical benefits?

Listing 3 illustrates one solution, the functional equivalent of Listing 1. The global **table** variable holds the offset addres-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ses of the procedures, **proc0**, **proc1**, **proc2**, and **proc3**. The first four statements in the main program initialize this array.

The inline procedure, **jump**, replaces the CASE statement in Listing 1, duplicating the action of the assembly language jump table in Listing 2. For reference, here's the assembly language for the inline code:

```
5B          POP BX
D1 E3       SHL BX,1
FF 97 table CALL table[BX]
```

The inline **jump** procedure takes a single word parameter, **index**, the counterpart to the CASE statement selector. Inline procedures (and functions) operate similarly to assembly language macros. When Turbo Pascal compiles the FOR loop, it replaces **jump** with the inline bytes. Just beforehand, the compiler also generates code to push any parameters onto the stack.

The first job for the inline code, then, is to pop the stacked **index** value into BX. Then, the SHL instruction multiplies BX by two, forming a 16-bit index into the jump table. Finally, the CALL instruction calls the procedure at the address previously stored in the table. (This example uses CALL where Listing 2 uses JMP, but the idea is the same.)

Listing 3 does not perform any range checks on variable **i**. In most programs, this is something you'll want to add. You must be certain never to use indexes outside the jump table range, or a program crash is certain.

Using Jump Tables

Before rushing out to perform your own jump table tricks, be aware that the **table** of offsets (Listing 3) must be a global variable stored in Pascal's data segment. The inline **jump** procedure assumes that DS addresses the data segment as DS normally does in Turbo Pascal. Declaring **table** local to a procedure or function will not work because local variables are stored on the stack, not in the data segment.

One disadvantage to the jump table solution is that you must use procedure calls as shown in Listing 3. You can't jump to other kinds of labeled statements because OFS can't calculate the address of a statement—only procedures, functions, and variables.

Are the advantages worth the trouble of using jump tables in Pascal? As a test (not listed here), I extended Listings 1 and 3 to call 21 procedures instead of 4. I removed the **Writeln** statements and then executed the FOR loop in each program 50,000 times on an 80386-based system. Listing 1 (with the Pascal CASE statement) took 23.72 seconds to run. Listing 2 (with the inline **jump** procedure) took 12.36 seconds, almost twice as fast.

As you can see, jump tables can dramatically improve the performance of Turbo Pascal's CASE statement. This isn't a trick you'll use all the time, but to squeeze extra speed from your programs, you might try giving your CASE statements a case of the jumps. ♦

Tom Swan is the author of Mastering Turbo Pascal *and* Mastering Turbo Pascal Files *published by Howard W. Sams, Inc., and* Programming With Macintosh Turbo Pascal *published by John Wiley & Sons, Inc.*

Code follows.

TURBO SOURCE SEARCH

TSS is a programmer's SIG that currently has 240+ disks representing 80MB of archived files!! TSS specializes in locating publicly accessible files in Turbo Pascal, BASIC, QBASIC, C, QuickC, ASM and Modula-2. We offer these files by mail as well as on-line.

TSS-BBS (617-545-9131 within the PC-Pursuit area) is an outlet for the programming code from *Programmer's Journal*. All files in the TSS SIG are freely accessible through the BBS to members and non-members. Members have increased time and access. A central library for publicly accessible programming utilities is a valuable resource, and we are certain that programmers recognizing that will support our efforts by becoming members.

TSS is also a BBS-by-mail. All files maintained by TSS-BBS are available to members for as low as $2.50/disk. All files on all disks are archived to maximize the number of bytes per disk. A yearly membership to TSS, $40.00, entitles you to purchase disks from the library for $4.00/disk. Bulk discounts can lower the price to $2.50/disk. We also offer custom filled disks. Send $4.00 for the catalog disk or $3.00 for a printed catalog. (Members receive first catalog disk FREE.)

TURBO SOURCE SEARCH
P.O. Box 11, Scituate, MA 02066
Voice: 617-545-6677
Data: 617-545-9131, 1200/2400 BAUD, 24 hrs.
VISA/MC/Check/Money Order/non-US extra

Number **35** on the Reader Service Card

So Help Me!™ **A NEW AND POWERFUL IDEA FOR CREATING POP-UP HELP WINDOWS!!**

Thanks to *So Help Me!* V3.00, adding context-sensitve help screens and menus to custom programs has never been easier! Simply put the text for help messages, menus, etc. in one or more ASCII files and use *So Help Me!*'s advanced Application Program Interface (API) to create multiple windows to display them.

FREE BONUS

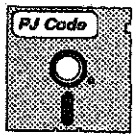
You'll be even more productive with *Show Me! Version III*, the $59.95 pop-up file viewing and pasting power tool sent **FREE** with your paid order!

FEATURES
- Up to four scrollable windows displayed simultaneously with full video attribute control
- Control returned to your program after a defined time period, when a predefined key is pressed, or when any key is pressed (the key value passed back for processing)
- All file I/O is handled including random file positioning by record numbers or by unique index marks to locate specific text
- Loads as a TSR program or as a child process; uninstalled on command; utilizes EMS memory; sample BASIC, C, Pascal and dBASE III+ driver programs provided
- Compatible with all languages and compilers for PC/MS-DOS 2.0 & later; requires as little as 20K

So Help Me! is only $89.95 (+$5 S/H).
To Order Call 800-634-3122.
Visa & MasterCard accepted • 90-day guarantee • No royalties
Serengeti Software • P.O. Box 27254 • Austin, Texas 78755-9954

Number **161** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



All code in this issue is available on 360K, MS-DOS disk, $10.00 postpaid. ORDER BY PHONE — VISA/MC orders call (503) 747-0800. Mail order information on page 72.

Code From "A CASE of the Jumps" by Tom Swan

Listing 1

```
program case1;
var i : integer;

procedure proc0; begin writeln(0) end;
procedure proc1; begin writeln(1) end;
procedure proc2; begin writeln(2) end;
procedure proc3; begin writeln(3) end;

begin
   for i := 0 to 3 do
      case i of
         0 : proc0;
         1 : proc1;
         2 : proc2;
         3 : proc3
      end { case }
end.
```

Listing 2

```
;-----  This program demonstrates how to write jump (case) tables
;       in 8086-family assembly language.  For MASM 5.0.

        TITLE   CASE.ASM
        PAGE    60,132
        DOSSEG
        .MODEL  small
        .STACK  256

        .DATA

;-----  Jump (case) table of addresses

table           dw      case0, case1, case2, case3

;-----  Various strings

prompt          db      'Enter 0, 1, 2, or 3: ', '$'
msg0            db      13, 10, 'Case statement #0', '$'
msg1            db      13, 10, 'Case statement #1', '$'
msg2            db      13, 10, 'Case statement #2', '$'
msg3            db      13, 10, 'Case statement #3', '$'
errmsg          db      13, 10, 'Data entry error', '$'

        .CODE
start:
        mov     ax,@data                ; initialize DS
        mov     ds,ax

        mov     dx,offset prompt        ; display prompt
        mov     ah,09h
        int     21h

        mov     ah,01                   ; get character
        int     21h

;-----  Test range of char in al.  Prepare bx equal to the case
;       statement number times 2, indexing the 2-byte addresses in
;       the jump table.

        mov     dx,offset errmsg        ; prepare for possible error
        cmp     al,'0'
        jb      endcase                 ; error if al < '0'
        cmp     al,'3'
        ja      endcase                 ; error if al > '3'

        mov     bl,al                   ; move digit char to bl
        and     bx,03h                  ; convert to value 0,1,2,3
        shl     bx,1                    ; multiply bx by 2
        jmp     table[bx]               ; jump to case 0,1,2 or 3

;-----  Each case is a separate routine, ending with a jump to the
;       end-of-case label.

case0:
        mov     dx,offset msg0
        jmp     endcase
case1:
        mov     dx,offset msg1
        jmp     endcase
case2:
        mov     dx,offset msg2
        jmp     endcase
case3:
        mov     dx,offset msg3

;----- Assume dx addresses ASCII$ message to display

endcase:
        mov     ah,09h                  ; display message at dx
        int     21h
        mov     ax,04C00h               ; end with exit code = 0
        int     21h

        END     start                   ; end of text, entry point
```

Listing 3

```
program case2;
var i : integer;
    table : array[ 0 .. 3 ] of word;

procedure jump( index : word );
   inline( $5B/ $D1/ $E3/ $FF/ $97/ table );

procedure proc0; begin writeln(0) end;
procedure proc1; begin writeln(1) end;
procedure proc2; begin writeln(2) end;
procedure proc3; begin writeln(3) end;

begin

   table[ 0 ] := ofs( proc0 );
   table[ 1 ] := ofs( proc1 );
   table[ 2 ] := ofs( proc2 );
   table[ 3 ] := ofs( proc3 );

   for i := 0 to 3 do
      jump( i )

end.
```

◆

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Turbo Professional 4.0

reviewed by
John J. Newlin

Regular readers of *PJ* will recall that I frequently mention the importance of programming tools to the successful design, coding, and testing of an end product. For example, I find Sammy Mitchell's Qedit editor (Version 2.0) and Eric Isaacson's A86/D86 (Version 3.12) indispensable. Many of us have had to develop our own tool kits along the way and perhaps have invested as much time and effort in tool development as in program development. But just because a set of code answers the mail with regard to providing program functionality does not mean that the code is a candidate for the tool box. To be an effective programming tool, a set of code must be sufficiently generic and flexible to be useful in a wide variety of applications.

This brings me to the point of this article. TurboPower Software recently released Turbo Professional 4.0 (Turbo Pro), a supercharged library of powerful Turbo Pascal 4.0 (TP4) routines that provide the programmer with a rich set of programming tools. TurboPower takes advantage of the new TP4 system of compilation to separate units by delivering the tools as modular components of a library of compiled TPU files. There are 25 units in all, and just these units alone would be well worth the price of the package, but there's more—much, much more.

In addition to the library of compiled units, TurboPower provides 14 executable demonstration programs, most of which were compiled using the supplied units. Several are very useful programs in their own right. But the real sweetener to this already rich pot is the source code. The complete, fully documented source code for every routine, including the demonstration programs, is also in the Turbo Pro package. Not just Pascal source code, but Turbo inline code, inline macros, and assembly source code are included as well. Now that, in my view, is quite extraordinary. Having access to the source code gives the programmer three big bonuses. One is that he or she can modify and customize any of the library routines to suit a specific purpose. The second is that studying the source code (it's quality programming) is a marvelous way to sharpen programming skills and gain an understanding of how some of the more complex code really works. The third bonus is that the programmer won't need an upgrade from TurboPower when Borland releases a new version of the compiler. When this occurs, all the programmer need do is recompile and link the source (and object, if needed) files to new units.

The Manual

Such a large, complex library would be difficult to document at best, but somehow the folks at TurboPower have managed to get their arms firmly around the beast and have produced an excellent user's manual. At 425 pages it's not a lightweight, but it is well organized into five main parts and quite easy to navigate. Each part is comprised of one or more of fifteen total chapters. Subprogram (procedure/function) names are bold headers at the top of each page, and the bottom of each page is flagged with the name of the unit containing the routine(s) described on that page. Every subprogram definition is accompanied by at least one code example, and in those cases where usage might be ambiguous, an incorrect example is included to complement the correct example.

Part 1 (Chapter 1) of the manual contains general information about Turbo Pro and how to use it. The license agreement is somewhat vague in that it does not address whether or not the purchaser is authorized to market executable programs produced using Turbo Pro routines. A call to TurboPower quickly cleared this up. An addendum to the READ.ME file will say that there are no restrictions on the distribution of executable files compiled using Turbo Pro modules. There are, however, restrictions on the distribution of the supplied source code and object code.

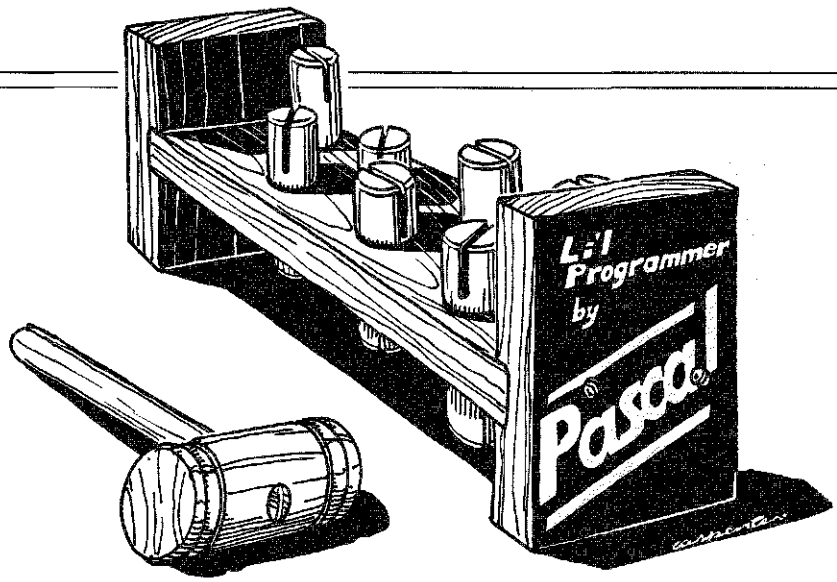
The Units

Now for the real meat and potatoes of Turbo Pro. Part 2 (Chapters 2–11) contains detailed descriptions for each unit in the library. The supplied library unit names are listed below.

TP8087	TPWINDOW
TPEMS	TPEDIT
TPRARRAY	TPASCIIZ
TPINT24	TPEARRAY
TPSTRDEV	TPOLDIFC
TPBCD	TPVARRY
TPSTRING	TPMENU
TPTSR	TPCMDLIN
TPEXTMEM	TPDOS
TPSCREEN	TPMACRO
TPSORT	TPERRHAN
TPCRT	TPINT
TPMACED	

The manual discusses TPCRT first, and it will probably be the first unit adopted and the unit most heavily used by the programmer. TPCRT is a standalone replacement for and a proper superset of the Turbo Pascal 4.0 CRT unit and contains a rich variety of screen handling extensions. Standard screen I/O is routed through TPCRT rather than DOS, and keyboard polling is designed to support TSR implementations. Some excellent cursor manipulation routines are present, including procedures for hiding the cursor, changing its shape, and restoring it. Also included are a number of powerful window control functions. I especially like Turbo Pro's **FastWrite** and **FastRead** procedures. **FastWrite** is an extremely fast string writing routine featuring complete con-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



trol over the placement of the string and video attributes used. **FastRead** reads text from designated screen coordinates into a string. TPCRT even provides detection of the currently active display type and, if installed, the presence of a second adaptor.

TPSCREEN and TPWINDOW perform screen manipulation functions on a more sophisticated level. TPSCREEN offers the programmer a means of writing applications that use a virtual screen—an area in memory (an alternate video page or a RAM buffer) that can be treated as if it were actual video memory. A virtual screen can be written to and then moved into video RAM. TPSCREEN provides all the necessary tools for creating and manipulating virtual screens (up to ten). TPWINDOW offers the true pop-up windowing capability that should be an intrinsic feature of any PC compiler. This unit contains routines that allow the programmer to make, display, and remove variously sized pop-up windows and even includes a **FastWriteWindow** procedure that automatically converts the specified window-relative coordinates to absolute coordinates.

Writing and debugging the TP4 code for a pull-down, pop-up menu system recently took me ten to twelve hours. Using MAKEMENU, an example program (actually, a full-featured utility) supplied with Turbo Pro 4.0, I designed and coded the same menu system in less than an hour. Very little testing was necessary since MAKEMENU allows you to test your menus as you design them. A library function that's included permits you to save menu structures in binary form for later recall and modification, and the program actually writes the TP4 source code for you. A listing of the source code for an example menu that I produced in just a few minutes with MAKEMENU appears at the end of this article. MAKEMENU itself is written using Turbo Pro units, and since the source is included, you can easily customize it for your own specific purposes.

Turbo Pro contains five related string handling units: TPSTRING, TPASCIIZ, TPEDIT, TPSTRDEV, and TPCMDLIN. TPSTRING contains over 35 general purpose string manipulation routines, the star of which is the **Form** function. **Form** is modelled on the TP3 BCD **Form** function and takes two parameters—a string template and a real number. **Form** returns a string formatted in accordance with the supplied template. For example, calling **Form** with a template of "$##,###,###.##" and the real number 17350.47777 will return the string " $17,350.48". A second version of the **Form** function found in the TPBCD unit supports that unit's high-precision BCD **real** type.

TPASCIIZ offers a solution if you need to manipulate strings with a length greater than 255 bytes. An ASCIIZ string terminates with a null (#0) character and contains no string length byte. TPASCIIZ provides a complete set of ASCIIZ string manipulation routines: **Concat**, **Delete**, **Pos**, **Length**, etc.

One of the best routines in the Turbo Pro package is **ReadString** from the TPEDIT unit. **ReadString** is a full-featured line editor with WordStar compatible line editing commands that give you complete control over accepting user string input. TPEDIT also includes routines for the input of **Real** (**ReadReal**), **Word** (**ReadWord**), and **Char** (**ReadChar**) data types. With TPEDIT routines, your software can more easily and safely accept user input. Each input routine includes a string parameter for a prompt and a string to present to the user for editing. For example, **ReadReal** will take a real number, convert it to a string for the user to edit, and return the edited number and a string representation of that number. Now that's what I call a useful, time-saving tool.

TPSTRDEV is a very small unit that exports two simple but powerful routines and one global variable. Here Turbo Pro takes advantage of the new TP4 capability to employ a custom text file device driver. The key to this unit is the export variable **TpStr**, which is type **Text**. Here's how it works. The following line of code is all that is needed to produce a properly formatted string:

```
Write(TpStr,'The value of pi is'
  ,Pi:2:5);
```

Executing this statement would produce the string "The value of pi is 3.14 ". The procedure **ReadStr** will load a normal string variable with the contents of the TPSTRDEV text buffer and then flush the buffer. The function **ReturnStr** accomplishes the same result but, unlike **ReadStr**, can be used to export the string buffer to other subprograms.

TP4 provides adequate high level program command line access through its intrinsic **ParamCount** and **ParamStr** routines. The Turbo Pro unit TPCMDLIN extends that access by adding routines that provide a much more extensive command line parsing capability. A good example is the case where you want your application program to accept command line arguments in the form of '/Switch nnn'. The routine to handle this is declared in the unit as

```
function GetArgNumber(var I :
  integer) : ;
```

Given a command line entry of "/R 3 /x /size $12", then

```
I := 1;
N := GetArgNumber(I)
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

returns N = 3, and

```
I := 4;
N := GetArgNumber(I)
```

returns N = 18, but

```
I := 2;
N := GetArgNumber(i)
```

sets an error code accessible from another TPCMDLIN function.

The TPDOS unit, which complements the TP4 DOS unit instead of replacing it, gives programmers a much broader range of access to DOS and BIOS functions. The main emphasis is on file handling and disk drive access. Among the many available are routines to count drives, change drives, read/write absolute disk sectors, get/set the DTA, and even to do random text-file access. TPDOS also contains **ExecDOS**, a sophisticated front end for the **Exec** procedure in Turbo Pascal 4.0 that temporarily compresses the heap to free as much RAM as possible for the executed program. More importantly, **ExecDOS** can be instructed to find and load COMMAND.COM in those cases where a secondary copy of the command interpreter must be present. **GetEnvironmentString** is a really handy routine residing in TPDOS. **GetEnvironmentString** is a string function that return a specified string from the DOS environment. For example, if you needed to acquire the location of COMMAND.COM, you would need only make a call like the following (where **ComSpec** is a variable of the type **String**):

```
ComSpec := GetEnvironmentString
  ('comspec=');
```

Upon return, **ComSpec** might contain "C:COMMAND.COM".

Turbo Pro provides a powerful set of interrupt management functions in the TPINT, TPTSR, TP8087, and TPOLDIFC units. TPOLDIFC gives the users of previous versions of Turbo Pro some capability to locate and exchange information with programs that use the old interface. TPINT is a true Interrupt Service Routine (ISR) management system. TPINT will get the desired interrupt vector, save it, install your ISR, and restore the original upon program termination. Not only that, but it will accomplish this entire sequence with just one call. For example,

```
if InitVector($16,MyISRid,@MyISR) then
  do_something_fun;
```

TPINT will manage up to 20 (!) ISRs and exploits the use of the TP4 interrupt procedure. Routines are included that allocate/deallocate needed stack space, switch stacks, chain to the original interrupt, and more. TPINT routines also properly restore all captured interrupts upon program termination—even if termination is effected by a runtime error.

A lot of programmers will just love TPTSR. This Turbo Pro unit contains a sophisticated, rock-solid Terminate and Stay Resident (TSR) management system that gives the programmer not only the tools to easily create memory resident applications, but also a means to learn and understand their inner workings by studying the supplied source code. With TPTSR you can define up to 8 pop-routines in any one program, define the trigger "hot key" combinations, and determine whether a routine is to be triggered through the keyboard interrupt or via an internal program call.

The low level interrupt handler code that supports this unit is extremely safe. At the trigger and prior to popping up, checks are made to ensure that DOS is not in a critical state and that no disk or BIOS I/O is in progress. Each defined pop-up is assigned its own stack, and the pop-up "engine" smoothly saves the current register state and switches stacks before calling the user defined routine. When loading, TPTSR saves the active interrupt vectors and properly chains to them after servicing the interrupts. The interrupts "trapped" and monitored by TPTSR routines are

05h - Print screen (software)
08h - System timer (hardware)
09h - Keyboard (hardware)
13h - Disk BIOS (software)
16h - Keyboard (software)
25h - Absolute disk read (software)
26h - Absolute disk write (software)
28h - DOS background task (software)

TP8087 is a unit containing routines needed for TSRs that use a numeric coprocessor for real number calculations. It's all great stuff, and there's even a routine to check for the presence of Borland's SideKick when loading your own routine and aborting if it is (SideKick should be loaded last).

SemWare Announces QEdit Advanced the "quick editor"

QEdit is one of the smallest, fastest, multi-file editors available for IBM compatible computers and has many features to enhance your productivity.

- Lightning fast, especially in file loading and saving, screen display, cursor movement, and global searching and replacing.
- Edit multiple files, as many and as large as available memory allows. QEdit can use all available memory, up to 640KB. Ideal for multi-file languages and all those .PRG files.
- Windowing allows up to 7 files or 7 views of the same file to be seen at once.
- You do not have to "learn" another editor with QEdit. QEdit's commands can be configured to any keys you like.
- Screen colors, tab settings, default file type and many other options can be configured the way YOU like it.
- Up to 99 named scratch buffers may be created and edited. Commands are available for easy cut and paste operations.
- Invoke a DOS "shell" or execute DOS commands directly from QEdit.
- Both line and stream block marking are supported.
- Keyboard macros allow automation of commonly used command sequences or text entry.
- Commands to shift an entire block left or right, to adjust indentation levels when using structured languages.
- Garbage stack provides restoration of the last "n" (user-settable) deletions.

Introductory Price
49.95
Mastercharge & Visa Accepted
Add $3.00 Shipping
Georgia Residents please include 5% sales tax

QEdit Advanced requires an IBM PC/XT/AT or Compatible, IBM PS/2 30 - 80, 128KB of available memory. Please specify disk format required.

To Order: Heritage Papers, Danielsville, GA 30633
Phone Orders: (404) 789-3226

Number 148 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

One of the included example programs is SMACS.EXE, a full-featured memory resident keyboard macro processor very similar to Borland's SuperKey. SMACS is built on the foundation of the TPTSR and TPMACRO units and it dramatically demonstrates the power and functionality of the Turbo Pro package. TPMACRO provides a set of low level tools that allow the programmer to define macros, assign macro names, dynamically record them, activate them, and save/retrieve them to/from a disk file. A related unit, TPMACED, implements a specialized macro editor that enables the user to insert, delete, and change keys in macro strings. Control over the maximum number of macros, the number of keys in each macro, and playback delay time is effected through constants defined in the TPMACRO unit.

Individual data array sizes in TP4 are still limited to 64K bytes, but Turbo Pro supplies three units—TPRARRAY, TPVARRAY, and TPEARRAY—that allow programmers to circumvent this limit. If you stop and think about it for a minute, you will realize that there are three areas in a DOS system that are candidates for data storage. They are normal RAM (which includes the TP4 heap), the disk, and expanded memory (EMS). As you might expect from their names, the three array units correspondingly support each of these three storage areas, as shown:

```
TPRARRAY - normal RAM
TPVARRAY - disk (virtual)
TPEARRAY - expanded memory
```

To facilitate program design and coding, each unit uses identical subprogram names for calling the supported operations. Minor deviations in the subprogram parameter lists necessarily exist as required by specific unit implementation, but calling convention similarity is such that the array units can be used interchangeably. More than one of these units may be used by a program at the same time by using the TP4 unit name override convention.

Each of the array units supports dynamic specification of the size and type of an array at runtime. The Turbo Pro manual explains that the arrays are zero-based (that is, the minimum index value for an array is zero) and provides some interesting benchmark information about array access overhead. Naturally, the virtual array unit, which swaps information to and from disk, is the slowest of the three. The manual also points out that data types larger than 64K take longer to access due to the requirement for Turbo Pro's routines to calculate a segment and an offset within that segment as a pointer to the correct array element.

BCD Precision

One of the first things that a business applications programmer learns is that the intrinsic Turbo 6 byte real does not provide the required accuracy for real number based financial calculations. That was precisely why Borland introduced the BCD (Binary Coded Decimal) variant in TP3. TP4 does not support BCD arithmetic, and unless the development and target machines have 8087 coprocessors, forget about BCD precision—unless you have Turbo Pro, that is. The TPBCD unit supplies the programmer with an extended set of BCD arithmetic routines. In fact, the BCD **real** type defined in this unit is implemented in the same internal format as that used in TP3, that is, a 10 byte array. This is one of several cases in which the unit routines are coded in assembly language to obtain maximum speed. If greater than 18 digit precision is required, the programmer can change a constant declaration in TPBCD and an equivalent constant in TPBCDLOW.ASM, the supporting assembly source file. TPBCD includes arithmetic, string and real number conversion, and transcendental function routines.

Error Handling

I was pleased to find that Turbo Pro includes a unit that allows an executing program to recover from nearly all runtime errors. TPERRHAN actually intercepts runtime errors before they are processed by the normal TP4 SYSTEM unit and passes error handling control to a routine defined in the unit. This routine calls a user defined error handler and then passes control back to the executing program or to the TP4 SYSTEM as appropriate. Three types of errors are unrecoverable and will be passed to the SYSTEM unit. Stack overflow, heap overflow, and any 8087 coprocessor error will crash the system. All others are recoverable, including I/O errors. To further bolster error handling, Turbo Pro includes the TPINT24 unit. The TP4 SYSTEM unit contains a

TRUE
MULTITASKING
With
MultiDos Plus
"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - * Intertask message communication. Send/receive/check message present on 64 message queues.
  - * Task control by means of semaphores. Get/release/check semaphores.
  - * Change priority-256 priority levels.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.
  - * and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: $24.95 *OR* $99.95
with source code

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

Number 109 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DOS critical error handler, but TP-INT24 provides a better defined set of error codes for a user defined critical error handler to act on. Error codes are delivered to the user via the **Int24-Result** function, which returns a word containing the TP4 **IOresult** code in the low byte and the DOS critical error code in the high byte. The following critical error codes are passed via the function:

01h - Invalid drive number
02h - Drive not ready
03h - Unknown command
04h - CRC error
05h - Bad request structure length
06h - Disk seek error
07h - Disk format not recognized
08h - Sector not found
09h - Printer not ready
0Ah - Write error
0Bh - Read error
0Ch - General failure
0Dh - Write protect error (mapped from $00)

Inline Macros

TPINLINE is a nice little goody locker containing a dozen or so TP4 inline macros. Inline macros are procedures or functions coded in Turbo inline format that instead of being called at fixed locations within the code envelope are inserted into the code by the compiler wherever they are declared. Their usefulness is assured by the fact that they can be called with parameter lists and will return function values, just as in regular Pascal subprograms. To get an idea of how to take advantage of this new TP4 feature, just browse the section of the Turbo Pro manual that describes the included macro routines. A peek at the source code supporting this unit will enlighten many programmers as to implementation techniques. An excellent example of a TPINLINE procedure is the one that implements a far call, where **ProcedurePointer** is a Pointer type:

```
ProcedurePointer :=
  @DoesVeryLittle; {points to
 a procedure}
.
.
{execute a far call to Does
  VeryLittle}
FarCall(ProcedurePointer);
```

Another good example is the function that normalizes a pointer. Normalizing a pointer is the process that adjusts the pointers offset value to be in the range of 0000h to 000Fh. This is neatly and quickley accomplished by the function **Normalized**, as illustrated below:

```
Var
  P1,P2 : pointer;  {these guys
    are pointers to anything}

begin
  P1 := ptr($0000:$0410);  {ini
    tialize P1 as unnormalized}
  P2 := Normalized(P1);
  writeln(HexW(seg(P2)),':',HexW
    (ofs(P2)));
```

The **writeln** statement above will produce 0041:0000. By the way, the **HexW** function above is one of several number base conversion routines from TPSTRING.

Two Turbo Pro units give the programmer the tools to manipulate non-DOS memory. TPEMS provides a number of standard EMS routines that support LIM 3.2 while TPEXTMEM deals with extended memory. Unfortunately, TurboPower has no plans to upgrade TPEMS to support the new LIM 4.0 specification. The reason they cite is valid and, at the same time, disconcerting. There just doesn't seem to be any real demand for LIM 4.0 software development tools, and the blame for that rests partly on the shoulders of the manufacturers. Take AST, for example. I still don't have the LIM 4.0 driver for my AST Premium/286, and if that's not a good reason for not writing software to ex-

Write Better Turbo 4.0 Programs...

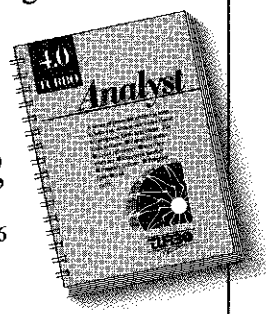
Or Your Money Back

You'll write better Turbo Pascal 4.0 programs easier and faster using the powerful analytical tools of **Turbo Analyst 4.0.** You get • Pascal Formatter • Cross Referencer • Program Indexer • Program Lister • Execution Profiler, and more. Includes complete source code.

Turbo Analyst 4.0 is the successor to the acclaimed TurboPower Utilities:

"If you own Turbo Pascal you should own the Turbo Power Programmers Utilities, that's all there is to it."

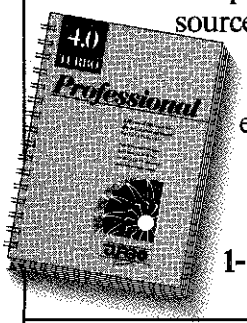
Bruce Webster, BYTE Magazine, Feb. 1986



Turbo Analyst 4.0 is only $75.

A Library of Essential Routines

Turbo Professional 4.0 is a library of more than 400 state-of-the-art routines optimized for Turbo Pascal 4.0. It includes complete source code, comprehensive documentation, and demo programs that are powerful and useful. Includes • TSR management • Menu, window, and data entry routines • BCD • Large arrays, and more.



Turbo Professional 4.0 is only $99.

Call toll-free for credit card orders.

1-800-538-8157 ext. 830 (1-800-672-3470 ext. 830 in CA)

Satisfaction guaranteed or your money back within 30 days.

Fast Response Series:
■ T-DebugPLUS 4.0—Symbolic run-time debugger for Turbo 4.0, only $45. ($90 with source code)
■ Overlay Manager 4.0—Use overlays and chain in Turbo 4.0, only $45.
Call for upgrade information.

Turbo Pascal 4.0 is required.
Owners of TurboPower Utilities w/o source may upgrade for $40, w/source, $25. Include your serial number. For other information call 408-438-8608. Shipping & taxes prepaid in U.S. & Canada. Elsewhere add $12 per item.

TURBO Power

TurboPower Software
P. O. Box 66747
Scotts Valley, CA 95066-0747

Number 25 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ploit LIM 4.0, I don't know what is. You can bet, however, that if the demand increases, TurboPower will be responsive.

No tool kit would be complete without a generic sort, and TPSORT exports three routines: an in-memory nonrecursive (for speed) quicksort, a routine for inserting a data item into the sort array, and a routine for extracting an item from the array. The **sort** function can sort up to 65K data elements of any type. Sorting is accomplished entirely in normal RAM, and paging to disk or EMS is not supported. Two things about TPSORT bother me. First, I see no way to accomplish a descending sort. Second, I don't think quicksort is the best choice among sort algorithms. Quicksort attains maximum sorting speed only when the data to be sorted is completely unordered. The more ordered the data is, the slower quicksort is. In fact, the plain vanilla bubble sort is faster than the quicksort on sorted data. A better choice, in my judgement, is the Shell-Metzner sort—it's faster sorting partially ordered data, and completely unordered data is the exception rather than the rule.

As I looked through this package, it occurred to me that it could benefit from the addition of a mouse support unit and a communications unit. It turns out that TurboPower includes a bonus disk with the Turbo Pro package. This disk contains 18 third party units and, of course, a mouse unit and a communications unit. Unfortunately, time and space limitations prevent my going through these units in any detail, but here are their names:

ASYNC4	EMUL87	INT24
STAKDUMP	TPSPL	BREAK
ENHKBD	LOCK4	TPHELP
TRIG	CHAIN	EXEUTIL
LPT	TPINFO	DATES
EXTEND	MOUSE4	TPPICK

Whew! There's a lot power and utility in the units I've discussed, and I've just scratched the surface. But there's more. Turbo Pro comes with ten utility programs and three demonstration programs. The utility programs have been compiled using Turbo Pro routines. Here's a list of the ten and brief descriptions of what they do:

SCMACS	SuperKey type TSR keyboard macro processor
PCALC	TSR programmer's calculator
PTIME	TSR programmer's time manager
PTRPT	Report generator for PTIME
PREF	TSR programmer's ASCII table
MAKEMENU	Interactive menu generator for TPMENU
BSORT	File sort utility
TFIND	Text find utility
REP	DOS command repeater
DIFF	Text file comparison utility

These fine utilities are fully functional, but they serve another, more important, purpose. The best way I know to improve programming skills is to study well designed source code. The source code for every one of these programs is included in the Turbo Pro package. You can "go to school" on the code and/or modify a particular program to suit your own purposes.

Continued...

If you see something here you like, by all means, compare it with what our competition has to offer.

Seidl Version Manager (SVM) Version 2.0 is a comprehensive source code configuration system designed for professional software developers. SVM tracks changes made to individual modules (revisions) and stores 'version' information that tells which revisions go into making which versions of a product. SVM supports multiple developers in both networking and non-networking environments. Some of its features include: revision branching, revision merging, user ids and optional passwords, seven different file access levels, a menu driven shell, formatted reports with user definable layouts, central journal logging, on-line help, optional data compression, a remote site update utility, and more. ***$299.95, LAN version $1000.00***

Seidl Make Utility (SMK) Version 2.1 is the most sophisticated product generation utility available. SMK is not just another copy of the of the UNIX make. SMK uses a structured, high level, *dependency definition language (DDL)* that is a programming language in itself. It supports symbolic constants, parameterized macros, for loops, if-then-else style conditionals, interactive statements, ambiguous path name expressions, and much more. In addition, SMK is lightening fast and won't bog down, even on your largest projects. ***$99.95, LAN version $500.00 (includes SMKgen).***

New!* SMKgen Version 1.0** is an automatic dependency generation utility. By examining the files that make up a product, SMKgen determines how those files depend on each other and amazingly writes a dependency file that can be processed by SMK. SMKgen can also construct link files and *filelist* files that integrate with SVM. With SMKgen, the task of placing a software project under SMK is reduced to executing a single command. ***$49.95

"SVM is a full-featured system that has all the essential capabilities you are likely to need, and is further distinguished by several important features that none of its competitors yet have." ***- The C Users Journal™, February, 1988.***

When you know the facts, you'll know we're the best.

For more information call: **1-313-662-8086**

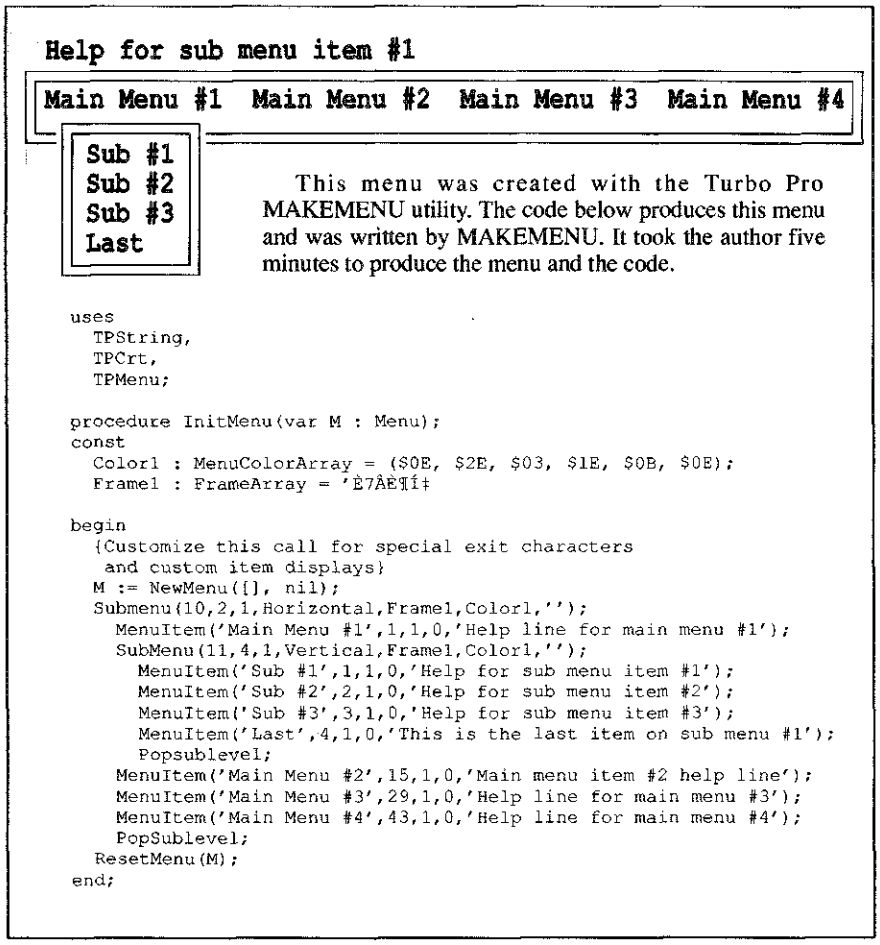
For a free copy of our competitive comparison report, write to us on your company letterhead. MC/VISA/COD accepted.

SEIDL COMPUTER ENGINEERING

3106 Hilltop Dr., Ann Arbor, MI 48103

Number **174** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED


Help for sub menu item #1
Main Menu #1 Main Menu #2 Main Menu #3 Main Menu #4
Sub #1
Sub #2
Sub #3
Last
This menu was created with the Turbo Pro MAKEMENU utility. The code below produces this menu and was written by MAKEMENU. It took the author five minutes to produce the menu and the code.

```
uses
  TPString,
  TPCrt,
  TPMenu;

procedure InitMenu(var M : Menu);
const
  Color1 : MenuColorArray = ($0E, $2E, $03, $1E, $0B, $0E);
  Frame1 : FrameArray = 'È7ÂÈ¶Í‡

begin
  {Customize this call for special exit characters
   and custom item displays}
  M := NewMenu([], nil);
  Submenu(10,2,1,Horizontal,Frame1,Color1,'');
    MenuItem('Main Menu #1',1,1,0,'Help line for main menu #1');
    SubMenu(11,4,1,Vertical,Frame1,Color1,'');
      MenuItem('Sub #1',1,1,0,'Help for sub menu item #1');
      MenuItem('Sub #2',2,1,0,'Help for sub menu item #2');
      MenuItem('Sub #3',3,1,0,'Help for sub menu item #3');
      MenuItem('Last',4,1,0,'This is the last item on sub menu #1');
      Popsublevel;
    MenuItem('Main Menu #2',15,1,0,'Main menu item #2 help line');
    MenuItem('Main Menu #3',29,1,0,'Help line for main menu #3');
    MenuItem('Main Menu #4',43,1,0,'Help line for main menu #4');
    PopSublevel;
  ResetMenu(M);
end;
```

As you may have gathered by now, I'm very high on TurboPower's Turbo Professional 4.0, and with good reason. Although I did not have the time to functionally test every unit, I could find no bugs in the ones I did use. At first glance, the price ($99.00) of this package might seem a bit high. After all, it costs as much as the compiler it supports. But considering the improvement in the quality of the programs you can write using Turbo Pro routines and the time saved in not having to write your own utility library, it's a bargain. ♦

John J. Newlin is a retired Naval Aviator who works for Hughes Aircraft Co. as a systems engineer. John has programmed extensively in assembly, Turbo Pascal, Marshal Pascal, and Modula-2 and is the author of SCOUT and ARCMASTER (John recently ported ARCMASTER to TP4.) John is a charter member of the Association of Shareware Professionals. Readers may write to John at the following address:

John J. Newlin
4060–228 Rosenda Court
San Diego, CA 92122

NEW

Write A Data Base Program
(end to end)
in 10 minutes in TPascal 4.0
using Turbo Programmer
(formerly Turbo GhostWriter)

Perfect for creating complex business applications!
Ideal for BASIC programmers who find TPascal too tough!
Spec your customer in the morning - Show a demo in the afternoon!

Features
- screen editor and painter
- automatic programmer documentation
- automatic data checking/validation
- plenty of "hooks" for customizing
- unlimited technical support

More features
- automatic B-tree indexing
- consistent user interface
- automatic context-sensitive help
- relational model to show customizing
- 30 day money-back gaurantee (less $14 shipping/handling)

No questions to answer. Just draw your screens the way you want them to appear, tell Turbo Programmer how to set up the indexes and that's it... Running 4.0 code in 6 seconds with no programming. Regular price $450.

Now Only $289 Orders & Info 800 227-7681
ASCII 3239 Mill Run, Raleigh, NC 27612-4135

Number **170** on the Reader Service Card

SOFTWARE INFORMATION SECURITY SYSTEM

Secure-D is a software-based product that enables you to limit or restrict access to your data files by cellular automaton random sequencer encryption prior to backup or transmission.

More than just a simple encode/decode utility. Parent/child operation permits retrieval of secured data by specific users. Optional compression function can be used with or without encryption to reduce the number of backup diskettes, phone charges or transmission times.

Secure-D for IBM and compatibles. $49.00 with a 30 day money back guarantee.

Az-Tech Software, Inc.
305 E. Franklin
Richmond, MO 64085
800-227-0644 or
816-776-2700

Number **187** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Heap Expander™ version 2.0

Now your programs can have virtually unlimited heap space using expanded memory, extended memory, disk space, or any combination of the three. And it's all transparent. The Heap Expander's initialization code checks the system's resources and uses whatever is available.

still $59.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:

- Turbo C
- Microsoft C 4.0 and 5.0
- Turbo Pascal 3.0 and 4.0

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

*Idaho residents add 5% sales tax
Foreign customers add $4.00 for shipping and handling.

The Tool Makers

75 Mt. Hermon Rd.
Scotts Valley, CA 95066
(408) 438-7412

Number **166** on the Reader Service Card

LAHEY SETS NEW FORTRAN STANDARDS

LAHEY PERSONAL FORTRAN 77 **$95**
Low cost, Full 77 Standard, Debugger, Fast Compilation

F77L FORTRAN LANGUAGE SYSTEM **$477**
For porting or developing, this is the critics' choice.
"Editor's Choice" . . . . . ***PC Magazine***
". . .the most robust compiler tested." . . . . . ***Micro/Systems***
". . .the most efficient and productive FORTRAN development tool for the DOS environment" . . . . . ***BYTE***

F77L-EM/32-bit $895 **F77L-EM/16-bit $695**
Break through the DOS 640K barrier. The most powerful PC Fortran Language Systems for downloading or writing large programs.

PRODUCTIVITY TOOLS
Profiler, ULI Mathematical Functions Library, Overlay Linker, Toolkit, Utility Libraries, Windows, Memory Boards, 80386 HummingBoard.

IF YOU DEMAND THE VERY BEST, THEN YOU SHOULD BE USING LAHEY. CALL US TO DISCUSS YOUR PC FORTRAN NEEDS.

CALL FOR NEW FEATURES INCLUDING MATH COPROCESSOR EMULATION

30 DAY MONEY-BACK GUARANTEE

FOR INFORMATION OR TO ORDER:
1-800-548-4778
Lahey Computer Systems, Inc.
P.O. Box 6091, Incline Village, NV 89450
TEL: 702-831-2500 TLX: 9102401256

Lahey Computer Systems Inc.

Number **44** on the Reader Service Card

MISOSYS

SOFTWARE FOR YOUR P.C.

Order Now 800-MISOSYS (647-6797)

ED/ASM-86™

- 80X86 Assembly Language development

An integrated editor, assembler, linker, and debugger for 8086, 80186, 80286, and 8087 assembly and disassembly. All features are in a single EXE file.

- a full-featured line EDITOR with intra-line editing, block move and copy, partial save and load.
- a full-screen turbo-type editor is supported on CGA-equipped systems.
- the ASSEMBLER supports structured coding; can directly generate a COM or EXE file; assembly to memory for faster debugging.
- The DEBUGGER includes symbolic disassembly of arbitrary code to disk.

ED/ASM-86 [M-86-030] . . $124.95 + $5 S&H

RATFOR-86™

- Rational FORTRAN translator

A professional implementation of RATional FORtran by James Beard, Ph.D. Provides structure and greater portability to FORTRAN programs to reduce your programming time and effort dramatically over that required when writing directly in FORTRAN. Includes tutorial reference manual.

- compiler has extensive macro support
- "arith" macro for binary arithmetic
- short form READ and PRINT
- enhanced switch case support
- Requires a FORTRAN compiler

RATFOR-86 [M-86-73] . . . $99.95 + $5 S&H

MISOSYS Inc.
P.O. Box 239
Sterling, VA 22170-0239
703-450-4181 MC, VISA, CHOICE
800-MISOSYS/647-6797

VA residents add 4½% sales tax.
S&H: Canada add $1;
Foreign use S&H times 3

Number **121** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Comparing CALL, GOSUB and Multi-Line Functions

by
Ethan Winer

With all the features and capabilities in the latest wave of BASIC compilers, many people have difficulty understanding the difference between a CALLed subprogram, a GOSUBed routine, and a multi-line user-defined function. In many cases it really doesn't matter which you use; however, some situations definitely favor one approach over the others. For example, using separately compiled subprograms is the only way to create a final program with more than 64K of code with QuickBASIC.

Essentially, a called subprogram would be used when you want to DO something or need to alter a variable. On the other hand, a function would be indicated when a value is to be computed based on the contents of one or more variables. Functions have the added characteristic that they cannot alter any of the incoming variables. Finally, GOSUB routines are useful when a large number of variables must be shared with the main program. Let's take a closer look at each of these methods in turn.

Subprograms

When a subprogram is called, the addresses of all the variables included in the parameter list are made available to the subprogram. Whenever one of these variables is assigned in the subprogram, the changes will also be reflected in the main program because the variable was actually altered. Further, any of the variables used in a subprogram that were *not* passed in the list are considered to be "private,"—that is, you can have a variable named **Price** within the subprogram and it will not conflict with a variable of the same name outside the subprogram.

Of course, you can also share variables between a main program and its subprograms with the SHARED statement. There are two ways to do this. One approach is to declare the variables as being SHARED within the subprogram. The only problem with this method is that in QuickBASIC, a bug causes the SHARED statement to be ignored once a program grows beyond a certain size. I have observed this with QB versions through 3.0, so the use of SHARED isn't recommended. Also, variables declared as SHARED within a subprogram will be shared only with the main program and not with any other subprograms.

The other method is to use DIM SHARED, and this will establish one or more variables as being shared throughout an *entire* program. Even though DIM is usually meant for declaring the size of an array, you can also use it to indicate which variables are to be global. Here's one example:

```
DIM SHARED Max, Position, Array$(250)
```

Functions

Functions are useful in a variety of programming situations and differ from subprograms in that they are not invoked by the BASIC CALL command. Rather, you simply refer to a function by its name as part of an expression. If you defined a function to return the square of a number, it might be written like this:

```
DEF FnSquare(X) = X * X
```

Then to square a number, you would simply include the function as part of an assignment, or perhaps use it in a PRINT statement:

```
Value = FnSquare(2.3)
```

or

```
PRINT FnSquare(A)
```

Functions also have the unique ability to return a numeric result based on a string, or vice versa. For example, BASIC's built-in STR$() function will accept a numeric value and convert it into the equivalent string form:

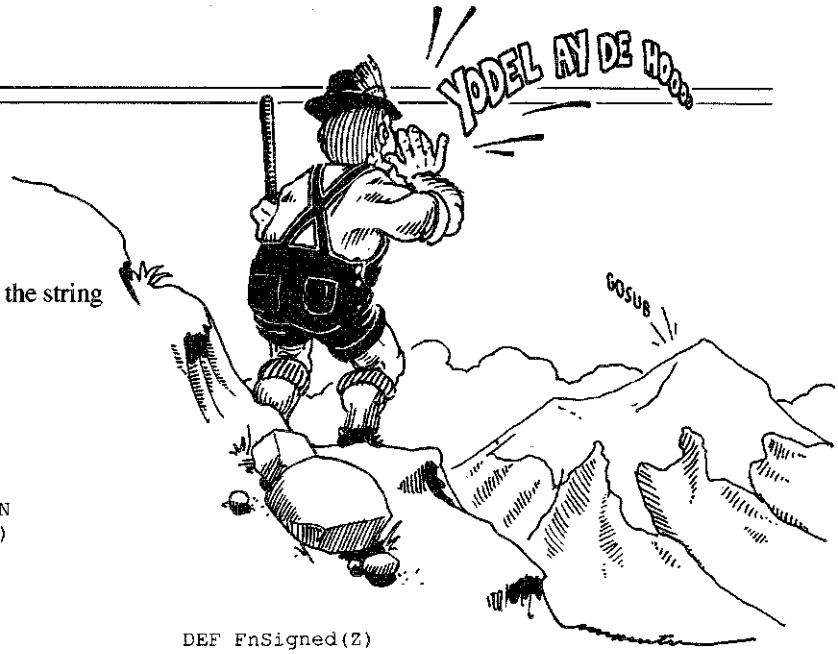
```
X$ = STR$(Value#)
```

Similarly, the VAL function accepts an incoming string, but returns its numeric value:

```
Number% = VAL("43562.89")
```

Since a function can't actually change any of the variables being passed to it, it would be less appropriate for, say, modifying a variable in some way. Of course, BASIC lets you do just about anything you'd ever want; however, using a function that way would require extra program instructions. For example, to convert a string to all upper case with the UCASE$ function available in Turbo Basic and QuickBASIC 4.0, you would have to assign a new variable:

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED




```
CustName$ = UCASE$(CustName$)
```

as opposed to calling a subprogram that capitalized the string directly:

```
CALL Upper(CustName$)

SUB Upper(X$) STATIC
   FOR X = 1 TO LEN(X$)
       Char = ASC(MID$(X$, X))
       IF Char > 97 AND Char < 123 THEN
          MID$(X$, X) = CHR$(Char - 32)
       END IF
   NEXT
END SUB
```

If you want to use a capitalized version of a string, though, without actually changing it permanently, a function is ideal:

```
IF UCASE$(Answer$) = "YES" THEN . . .
```

or

```
PRINT UCASE$(CustName$)
```

This way, the string isn't actually altered, but your program can use it as if it were. Because of this behavior, functions are not often used to modify or print variables, or to perform file access, though of course they could be.

Functions are also ideal for providing a single result based on a number of input variables. Many of BASIC's built-in functions work like this as well; for example, INSTR accepts two different string arguments and returns one numeric result. Another place a function might be useful is where you need to find the maximum or minimum of two or more values:

```
DEF FnMax(X, Y)
   IF X > Y THEN
      FnMax = X
   ELSE
      FnMax = Y
   END IF
END DEF
```

Then to print whichever variable is greater, simply include the function within a PRINT statement:

```
PRINT FnMax(Value1, Value2)
```

In subprograms, the incoming variables may be freely changed, whereas BASIC makes a temporary copy of any parameters being passed to a function. Then the addresses of these copies, rather than the addresses of the original variables, are made available to the function. Thus, any variable assignments within a function will be in effect only while the function operates since only the copies are being manipulated.

In the function below, even though the incoming variable Z is being modified, the assignment will not be reflected in the main program once the function has finished.

```
DEF FnSigned(Z)
   IF Z > 32767 THEN Z = Z - 65536
   FnSigned = Z
END DEF
```

Another important point is that unlike in subprograms, in BASIC, variables that are not part of the function's incoming parameter list will be common to the rest of the program. Of course, functions can also make use of local variables, but they must first be declared STATIC.

Which brings up an important point. Even though STATIC is generally used to declare a variable as being private to a function or subprogram, its *real* purpose is to permanently allocate storage space while a program is being compiled. Normally, a programmer uses a high-level language to avoid having to deal with the messy details of memory allocation, DOS functions, and so forth. In this case, however, it is important to understand how variables are stored.

When a variable has been declared as STATIC, an area of memory is set aside to hold it at the time the program is compiled. This is exactly the same way in which regular variables are handled in a main program. But with the introduction of Turbo Basic and QuickBASIC 4.0 and their support for recursive functions and subprograms, you may now stipulate that variable space is to be created when the program is run. In this case, temporary space is set aside on the processor's stack to hold the variable's value.

In a recursive subprogram, any variable that is *not* declared as STATIC will be created and initialized to zero (or a null string) each time the subprogram is invoked. But in some situations you might want to retain the value of certain variables between successive calls. This is precisely what the STATIC statement is for—it creates a permanent memory location in normal RAM for those variables. Otherwise, the variables exist only while the subprogram is running and the stack memory that holds them is abandoned when the subprogram ends.

This is an important distinction because recursive functions must preserve all of their local variables if they are to be repeatedly invoked. If a recursive function or subprogram is midway through a series of calculations and then calls itself, the second invocation must not destroy or overwrite any variables from the level above. This is why new memory areas must be found to hold the non-STATIC variables at the time

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

	BASICA/GWBASIC	Turbo Basic 1.1	QuickBASIC 3.0	QuickBASIC 4.0
GOSUB				
DEF FN				
SUB				
FUNCTION				
CALL INTERRUPT		1	2	2
INLINE .ASM				
LINKABLE .ASM				

notes: 1 built into the language
2 must link separate .OBJ file

A comparison of procedure capabilities in several popular versions of BASIC

the function is used, rather than once at the time the program is compiled.

GOSUB

In most cases, a GOSUB routine is not as useful as a called subprogram since the name of each variable being manipulated must be whatever the subroutine is expecting. In the old days, the only way you could have a subroutine do the same thing to many different variables was to constantly assign and re-assign:

```
Temp$ = CustName$ : GOSUB 1200 : CustName$ = Temp$
```

One exception, though, is the case in which a large number of variables must be shared between the main program and the subroutine. When I first began writing a word processor I've been tinkering with for about a year, I started to write the margin setting routine as a subprogram. But then I was faced with either passing all of the margins and other assorted default values as parameters or else declaring the twenty or so variables as being SHARED. It finally hit me that a plain old GOSUB made the most sense—the routine could access all of the variables it needed, and I didn't have to pass or share any of them.

Yet another advantage of GOSUB is due precisely to the fact that it is not kept separate from the rest of the program. One of QuickBASIC's many restrictions in its error handling is that errors that occur in a subprogram must always be handled in the main program. If the routines that perform disk reading and writing are all in the main program to begin with, you'll have less difficulty resuming a program's execution where you want to.

Functions in QuickBASIC 4.0

QuickBASIC 4.0 introduces yet another type of function, which Microsoft refers to as a *Function Procedure*. This type of function is really just a variant of the subprogram though it does return a value as part of an expression. As in subprograms, variables not declared STATIC in a Function Procedure will be local to the procedure. Also as in subprograms, any of the incoming variables may be freely altered by the function. Further, this type of function does *not* need to be physically positioned before the lines that invoke it in a program as required by the original DEF FN functions.

It is also possible to call a Function Procedure (or subprogram) in such a way that the incoming variables may *not* be altered if you prefer. This is done by enclosing the variable's name in parentheses, which QuickBASIC will then treat as an *expression*. Even though this may seem a little odd, it makes perfect sense once you think about it. For example, if you have a subprogram that expects an incoming numeric variable, you could also call it with a numeric expression like this:

```
CALL DoSomething(X * 5)
```

In this case, it would not be appropriate for BASIC to pass the address of the variable X since what you really want is to send it the result of X after it has been multiplied by five. Therefore, QuickBASIC first performs the calculation, then stores the result in a temporary location, and finally sends the address of the temporary location to the subprogram.

In a similar manner, you can trick QuickBASIC into thinking a variable is really an expression by enclosing it in parentheses:

```
CALL DoSomething((X))
```

Microsoft's otherwise excellent QB 4.0 documentation mistakenly describes this technique as "passing by value," claiming that the *value* of the variable is being passed rather than its address. In truth, though, a temporary copy of the variable is created and the address of the copy is then passed to the function. Thus, any changes made to the incoming variable will affect only the copy, and the original variable's value will be preserved.

Summing Up

We have looked at a variety of different ways that blocks of BASIC code can be organized. Even though functions now have the capability to extend across multiple lines and do operations like printing and file I/O, they are more often used to perform calculations based on variables or values.

And while subprograms can be used to do nearly anything you'd like, they are better suited to manipulating variables or performing specific actions. Finally, GOSUB routines provide a simple way to share many variables with the main program, as well as to simplify error and event trapping. ♦

Ethan Winer owns Crescent Software in East Norwalk, Connecticut, and is the author of the popular QuickPak utilities for QuickBASIC and Turbo Basic.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

80286/80386 Stuff

LMI UR/FORTH-386
LMI OS/2 UR/FORTH
LMI TMS34010 FORTH Cross Compiler
Laboratory Microsystems Incorporated
P.O. Box 10430
Marina del Rey, CA 90295
(213) 306-7412

If you need some fast new threads, Laboratory Microsystems (Ray Duncan) has added 286 and 386 Forth interpreters to their language products. LMI has released UR/FORTH-386 ($350), a high performance 32-bit Forth programming environment that runs in the 80386's native 32-bit protected mode under Phar Lap's DOS Extender (2 megabytes of memory required). UR/FORTH-386 allows programmers to take full advantage of the enormous address space and enhanced instruction set of the 80386. The only limit on the size of application code and data structures is the amount of RAM that can be installed in the machine.

Major features of UR/FORTH-386 include a segmented memory model, object-compiled nucleus, direct threaded code interpreter, and fully hashed dictionary. The dictionary, headers (symbol table), and hash table are in separate segments that can be resized dynamically as needed (up to 4 gigabytes if your PC has that much memory). The development environment also sports a table-driven, full screen, multi-file editor, a table-driven 80386/80387 assembler, relocatable binary overlay capability, and an enhanced user interface. UR/FORTH-386 supports all text and graphics modes of the MDA, CGA, EGA, and Hercules video adapters (the VGA is currently supported in EGA modes). Graphics support includes a set of graphics drawing routines coded in native 80386 assembly for line drawing, arcs, ellipses, circles, region filling, bit-block moves, custom character sets, text in graphics modes, clipping, and pixel reading and setting. LMI UR/FORTH-386 has separate library support for both the 80287 and 80387 coprocessors.

LMI has also released OS/2 UR/FORTH ($350) and a TMS34010 FORTH Cross Compiler ($1000) for the PC. LMI Forth packages come with a detailed 500+ page manual with abundant source programming examples for Forth utilities, hardware interfacing, graphics, and windowing. All LMI Forth language products are Forth-83 compatible (both 16- and 32-bit versions). LMI provides technical support for its Forth programming products on a dial-up bulletin board available 24 hours per day that supports 1200, 2400, and 9600 baud. Full time students and faculty are eligible for a 25% discount on all LMI language products.

Lahey FORTRAN F77L-EM
Lahey Computer Systems, Inc.
P.O. Box 6091
Incline Village, NV 89450
(702) 831-2500

The latest FORTRAN release from Lahey Computer Systems is F77L-EM Version 1.10 ($695), targeted for 80286 or 80386 PCs with extended memory. F77L-EM generates 80286/287 code operating in the 80286's protected mode, supporting applications up to the 16 megabyte limit of the 80286. The executable code is designed to run under the A.I. Architects OS/286 or OS/386 DOS extender runtime environment ($195, see *PJ*, 6.2). This allows FORTRAN applications to break the 640K barrier for code and data size while still running under DOS. F77L-EM is bundled with a runtime version of OS/286 and OS/386, an extended memory linker LEM, a BIND utility (to bind the OS kernel

Continued on page 74 . . .

PLOT TEXT ON
ANY GRAPHICS SCREEN!!

YES, we said ANY graphics screen, even VGA! ***Finally!*** **Xgraf** is a super set of low level assembly graphics routines that you call directly from Compiled Basic. The **Xgraf** kernel replaces Basic's confusing statements with consistent full featured calls specifically designed for the Basic programmer. **Xgraf** supports **Hercules, CGA, EGA, EEGA, VGA, and user defined screen sizes.** All modes can be accessed in real time - no recompiling!

Advanced features extend Basic graphic capabilities!

In addition to Basic's graphics features, **Xgraf's** ***advanced features*** include zooming, tiling, a virtual screen page, screen compression to disk, windowing, screen importing, *and much more!*

Finally! **Xgraf is only $99.00 + $4.00 s&h**
(Xgraf requires the Microsoft QuickBASIC compiler)

We *specialize* in libraries and tools for Compiled Basic. Our catalog features the ***Finally!*** family of programmer's libraries and *other top flight tools.*

KOMPUTERWERK - YOUR BASIC RESOURCE

KOMPUTERWERK
851 Parkview Blvd.
Pittsburgh, PA 15215

Call: 1-800-423-3400
PA & AK call (412) 782-0384
VISA and MC accepted!

Number 155 on the Reader Service Card

C++

FULL COMPILER – not a preprocessor

- Complete software-development system
- Selectable AT&T C++, ANSI C, K&R C
- Based on proven Oregon Software technology
- Generates extremely fast, compact code
- Comprehensive error checking
- Backed by responsive support engineers
- Strongly typed language

DATA ABSTRACTION FACILITY (classes)

- Operator overloading
- Information hiding/sharing
- Constructors/destructors

✓YES! I want top performance at an affordable price

To order, or for more information, call 1-800-874-8501
6915 SW MACADAM, SUITE 200, PORTLAND, OR 97219

OREGON SOFTWARE
Professional Products for Software Development

Number 102 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

T-SHIRTS

Multi-color imprints on the highest quality 50/50 cotton T-shirts.

$12.00 each —includes shipping & handling

Available in sizes **S, M, L, XL & XXL**

PARALLEL PORTS

BORN TO PROGRAM

COMMON C BUG

RS-232 HEAD

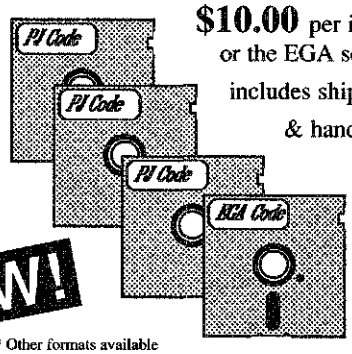
Bugs Rule

New— 4 color imprint on a white shirt

NEW!

CODE DISKS

The code listings from ***PJ*** Volume 5.5 forward, and from Michael Abrash's EGA series are now available on easy-to-order individual issue diskettes. 360k MS-DOS disks standard.*



$10.00 per issue, or the EGA series, includes shipping & handling.

* Other formats available

10% discount on any order of 10 or more T-Shirts, Disks or Back Issues. **ALL ORDERS MUST BE PREPAID.**

Use the order form on the card opposite. Or save time— order by phone using your VISA MC— CALL: (503) 747-0800

BACK ISSUES

Supplies are Limited—Order Today!

Vol. 1.1: Norton batch files and Pascal I/O; Program style manual; COBOL on the IBM PC; Benchmarks; System analysis; BASIC modules; Rounding error.

Vol. 1.2: Defining character sets; C86 BIOS calls; Troubleshooting; Copy protection; Pascal code; Input routines; Writing an assembler.

Vol. 1.3: ASM dumb terminal; ASM tutorial routines; C source code; ASM graphics line drawing; Text editors reviewed; Utilities; Self-documenting programs; Text files to dBASE II; dBASE II PEEKS & POKES.

Vol. 1.4: Diskette directory via Pascal; MASM WHILEWEND; C/ASM interface; ASM stay resident routine; Structured programming; Software law; Pascal tutorial; Default values and variable names.

Vol. 1.5: Multi-tasking software review; Transporting C programs; Programmer's calculator; dBASE II bargraphs; Utilities; ROM BIOS equates; Pascal check program; Combining modules in Assembler; Structured programming in BASIC; Agreement outline.

Vol. 1.6: ASM printer configuration; DOS patch utility; Hi-speed graphics; Software ownership; Structured programming revisited; C and BASIC compared; Software protection; dWINDOW review.

Vol. 2.1: Freelance consulting; Sorting in C; Pascal text editor source code; CRT distress; Compiled BASIC; Assembly modules for C; Designing a relational database; Ultra Utilities; Punctuation and Style reviewed.

Vol. 2.2: Interview with Brian Kernighan; Understanding C through assembly language; Utility routines in C and assembly; Music in C; Automatic line number optimizing in BASIC; Assembly language tutorial; Full-screen input routines in BASIC; dBASE II printer control codes; Pascal tools reviewed; C programming support products.

Vol. 2.3: Software development environments; Creating a productive screen handling tool; Assembly tutorial; Generating graphics within windows; Composition and editing system for musical ideas; Hybridizing assembly and BASIC; Creating screens; Linking Assembler to Pascal and BASIC.

Vol. 2.4: Writing a custom ROM BIOS for PC DOS; RISC, Transputers, and mass storage; The future of disk protection; 8088 assembly macros; Random access files in BASIC; C utility routines you can buy; Using the PMATE macro language; Choosing a database manager; E/SP syntax-checking; Pascal editor reviewed.

Vol. 2.5: Developing the APX CORE Executive; From assembler to C via a 211 byte .EXE main( ); Scroll control using BASIC and Assembler; Faster screen handling with BASIC and ASM, Random access files, Part 2; RTOOL, a PC-resident programmer's toolkit; dBkey reviewed; Pascal's pointer power.

Vol. 2.6: Periscope--building a hardware board for the IBM PC; Intel 80286 bugs by FNE; AT&T surprises all; Field Usage and pointer subtraction in C; Calendar functions for BASIC; Memory segmentation; Reduce your .EXE file sizes; Enhancing Debug; dBASE II to dBASE III? THINKTANK reviewed.

Vol. 3.1: November's Comdex; C--its uses and abuses; Using the C preprocessor; Memory segmentation examples; Index files, Part 1; Structured programming language; Periscope, a symbolic debugger.

Vol. 3.2: UNIX! (again); Artificial Intelligence; Doctor C's tips--pointers to functions; Calendar date routines in BASIC; Calculate record layouts, automatically!

Vol. 3.3: NAPLPS and the professional programmer; Instant-C; Printer control from a batch file; Repositioning the stars in Starfinder; Reducing EXE file size through data positioning; tools for better programming.

Vol. 3.4: More UNIX stuff; C programming books, Part 1; C code excerpts, Part 1; How to cure the "dreaded locked keyboard" condition; Calling MS DOS from Turbo Pascal; DOS 2.x sort and find filters; the SET Tools.

Vol. 3.5: Windows done here; C programming books, Part 2; C code excerpts, Part 2; A menu-driven PC using DISPLAY.COM; Turbo Pascal and MASM: making them work together; dBASE II routines; Turbo Pascal 3.0; TopView Programmer's Toolkit.

Vol. 3.6: About C's HLL overhead; How to recover all your costs; Report on the Comdex and Usenix shows; The Santa Cruz Operation XENIX for the IBM PC; Common C bugs; CTOOLS2; Turbo Pascal 3.0 and subdirectories; More dBASE II routines; Clipper reviewed; A better way to interface BASIC and Assembler.

Vol. 4.1: FNE on PC/AT hard disk problems and plunging 64K DRAM prices; AUTOINC explained; Sorting using an AVL tree; Combining binary trees; Using DOS to call programs from Turbo Pascal; An operating system for the people; Final installment of Brett Salter's dTOOLS; Two microcomputer graphics books reviewed.

Vol. 4.2: FNE on Intel 80386, single- vs. multi-user, multi-tasking computing, benchmarks; The home office syndrome; Frank Greco's industry news; Funky string functions; Capturing file handles in Turbo Pascal; Proposed ANSI C Standard, Part 1; Optimizing for speed in Assembler; Memory resident control over print screen functions; Structuring BASIC output with field statements; QuickBASIC Compiler and networking; Programming for Printer Output reviewed.

Vol. 4.3: FNEanother RISC machine, the paging game, software viewpoint, programmer talent, and Aloha, 32032; Sun Microsystems' next generation PC; Consulting—reusable modules; INT24.PAS, a critical error handler for Turbo Pascal; Memory model independence in Assembler; An approach to windowing for the assembly language programmer; Proposed ANSI C Standard, Part 2; Microsoft and Lattice C 3.0 reviewed; Accelerator boards for software development reviewed.

Vol. 4.4: Hal W. Hardenbergh on adult toys, tomorrow's micros, HALGOL update, benchmarks, and RISCs; Frank D. Greco on parallel computers, RISC, UNIX; Consulting—efficient custom programming; DOS 2 file handles and file I/O; Programming the WordStar Document File Conversion Utility; More Optimizing for Speed; Teach an old compiler a new trick; Memory resident programs in C; Tips on formatting with BASIC PRINT USING statements; the structure of dBASE II and III file headers.

Vol. 4.5: Hal W. Hardenbergh on 32-bit PCs, the 68040, Weitek's data-path part, etc.; Frank D. Greco on large screen monitors, language translators, UNIX, and more; Prolog intro, Turbo Prolog reviewed; Consultingtax traps, consulting styles, nest eggs; ASM—Search and Display; ASM—the GROUP statement; The proposed ANSI C Standard, Part 3; C—measuring stack usage and Lattice C's startup code; Short boards reviewed.

Vol. 4.6: DTACK GROUNDED; Unix Books, Part 1; How many amoebas at one o'clock; DOS functions 45H and 46H; DOS Terminate ans Stay Resident; Turbo Pascal inline code; Pascal—Assembler souce code for DOS function calls; Alignment problems in C and Managing memory allocation; Proposed ANSI C Language Standard, Part 4; The Canon A1 laserprinter; Geniferd BASE applications generator reviewed; Advanced Trace86 debugger reviewed.

Vol. 5.1: DTACK GROUNDED; The new DOS; Marketing and rules to live by; File redirection; Inside the EGA; String allocation in Pascal; ANSI C Language Standard, Part 5; More tips on formatting BASIC data.

Vol. 5.2: DTACK GROUNDED; Starting your own software house; Pink noise; TSR graphics print screen utility; Coding guidelines for resident programs; Parallel processing with the EGA; C—portability in practice and using header files; Proposed ANSI C Language Standard, Part 6; Indexing in BASIC; A first look at Turbo C.

Vol. 5.3: Starting your own software house--publicity; Unix books, Part 2; Consulting—merger mania; EGA data control; Modula 2 for Pascal lovers; Optimizing Turbo Pascal; ANSI C language standard; Coping with ANSI C and forcing file updates; Formatting BASIC output with buffered PRINT USING.

Vol. 5.4: Starting your own software house—technical support; Unix books, Part 3; Consultant productivity and effectiveness; EGA set/reset capabilities; ANSI C language standard; Character tables and string functions in C; MWWIND—a menu with windows utility; Variable passing and hiding in QuickBASIC 2 subprograms.

Vol. 5.5: Intel's 80386 vs. OS/2; Metagraphics President Jack Davis interview; Consulting—making computer tools pay for themselves; ASM—the invisible CapsLock; TSRs using expanded memory with Turbo Pascal; Note on optimizing Turbo Pascal; Proposed ANSI C Language Standard; Printer output from Turbo C programs; Matchup of Turbo Basic 1.0 and QuickBASIC 3.0.

Vol. 5.6: Multics, OS/2, paging and segmentation; The Usenix conference and other news; How to get rich in consulting; Shell in Marshal Pascal; Turbo Pascal 4.0 preview; Handling fixed length charaacter arrays in C; The proposed ANSI C Language Standard.

Vol. 6.1: Consulting is an attitude; Write mode 3 of the VGA; The LIM 4.0 expanded memory specification; Environment variables in Turbo Pascal 4.0; A timer program in C; Porting mainframe software to the PC--FORTRAN compilers under DOS and the Defincon 32-bit coprocessing engines.

Vol. 6.2: Hal Hardenbergh on RISCs, CISCs, and Fabs; Pink noise; Yet another VGA write mode; Inside Turbo Pascal 4.0; The proposed ANSI C Language Standard; Converting a Turbo C program to a TSR; 80286 and 80386 programming; 80386 software development; The Norton Guide for BIOS/DOS Assembly.

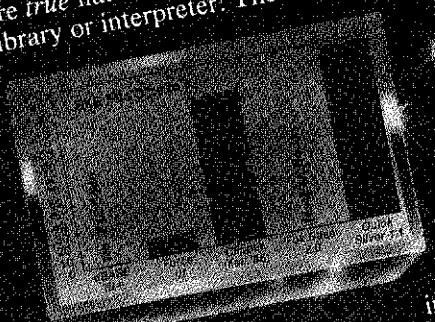
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FORCE III

The Ultimate dBase Compiler

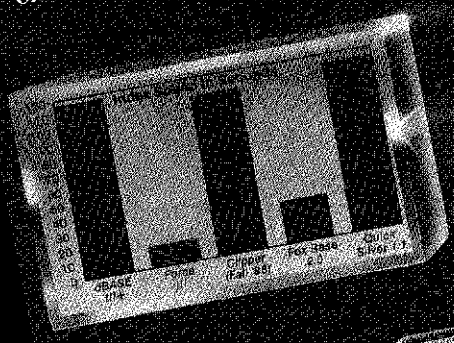
ORCE III is the world's fastest dBase
ompiler. It compiles and links the quickest,
nd the .EXE files it creates run so fast
you'll swear you wrote your program in C.
To give you an idea how fast, take a look at
the *Index Speed* graph below. This graph
compares indexing speeds of the different
products using a database which contained
8,000 records. The database was indexed
on the numeric field, NUM.

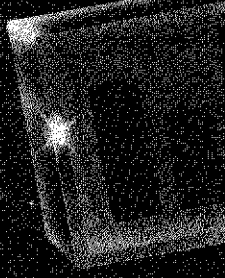
Best of all, FORCE III creates very small .EXE files. The .EX
are *true* native code files and do not rely on any kind of memory
library or interpreter! The reasons the .EXE files are so small a
simple: FORCE III is a
"intelligent" compiler a
uses those routines fro
library that your progra
and secondly, the enti
written in Assembler,
makes the routines ve
and very fast.



In the *Index Size* grap
index files created by ea
compared. As you can s
FORCE III the fastest at i
creates the sma

(Tests conducted on an IBM 8Mhz. AT wi





$129

Royalty Free and fully linkable!

In addition to being the fastest true dBase compiler on the market, FORCE III generated .EXE files carry no royalties whatsoever! And FORCE III compiled programs can be linked to Borland's Turbo C and other C compilers using the Intel/ Microsoft OBJ format. And linking to Assembler is a snap. We even included an ASCII graphics generator for easy screen and menu generation.

Arrays, UDFs, sound, new functio

Force III includes features you've come to
dBase compilers. For the programmer we have
FOR-NEXT loops, C like file primitives, a
even included some Clipper compatible c

Call now, 800-922-3001

FORCE III comes with extras,
excellent reference ma
customers, we'll incl
Advance dBASE
Technique
value). All th
s&h). In C
Europe call
write: Sopho

Products mentioned above

PRODUCED 2014 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

into an EXE file), and a source code symbolic debugger.

The F77L-EM compiler runs under the real mode DOS and is compatible with Lahey's DOS FORTRAN-77 compilers but has an option to generate 80286/287 code. The object modules created are designed to be linked with LEM, an extended memory linker, and run on an 80286 or 80386 machine or the A.I.Architects Hummingboard 80386 Coprocessor (*PJ*, 6.2). Lahey's F77L-EM compiler is fast, comparable to their DOS compilers, and supports the FORTRAN-77 ANSI standard with several useful extensions. The compiler supports a variety of command line options and accepts both standard and free format source files. Lahey extension keywords provide direct interface support for both Microsoft and Lattice C compilers. F77L-EM comes bundled with a source code debugger (SOLD) that uses a debug file generated by a compiler option and runs in the extended memory environment.

Consider F77L-EM if you are porting mainframe software. I was able to compile some large (by PC standards) mainframe FORTRAN programs with relative ease and run them on both ATs and 386 machines with extended memory. The manual is 300+ pages with a separate brief section on the OS/x86 runtime environment and the LEM linker. The manual is well written, with FORTRAN-77 extensions distinguished by a shaded background and a separate appendix listing all extensions. Technical support is available via phone or a bulletin board.

Soft-ICE Version 1.0
Nu-Mega Technologies
P.O. Box 7607
Nashua, NH 03060
(603) 888-2386

Have bugs got you down? Need a refreshing change? Soft-ICE ($386) is an unusual debugger for MS-DOS that provides hardware-level debugging capabilities. Soft-ICE runs only on 80386 machines and uses the 386 protected mode to run DOS in a virtual 8086 machine. Soft-ICE uses protected mode features of the 80386 including memory paging, I/O privilege level, and the 386 hardware breakpoint registers to considerably enhance DOS debugger capabilities. The 80386 has four hardware debug registers that offer powerful debugging capabilities. Each of these registers may be set to a specific address (and length up to 4 bytes) and to breakpoint on instruction execution, data writes only, and data reads or writes. The 4K memory paging capabilities allow breakpoints on specific memory ranges (page faults). The 64K I/O space can be protected by the I/O privilege level and I/O permission bitmap of the protected mode 80386 while interrupts are vectored through interrupt gates. When running with extended memory, Soft-ICE provides a full 640K for DOS. The debugger can be used standalone or in conjunction with DOS debuggers such as CodeView or Periscope.

Soft-ICE can be set to break on memory locations (up to 4), on up to 10 memory ranges, on instruction executions, on specific I/O ports, or on interrupts (hardware or software). Most breakpoints can be qualified by a count. Memory address, interrupt, and I/O breakpoints can be further qualified by equal or not equal to a specific value. In addition, qualifiers can include multiple breakpoints. Soft-ICE also supports a soft boot capability to allow debugging with non-DOS operating systems or self-booting programs. It can be used to debug loadable device drivers, interrupt handlers, or boot sequences—a very difficult task for traditional debuggers. Soft-ICE can be loaded as an executable program or as a device driver in order to share memory with VDISK or other extended memory programs. The program's window can be popped up by breakpoint or a user definable keystroke (typically Alt-D), and the program features online help. Soft-ICE supports the normal debugger commands to unassemble and read and write to memory and I/O ports. Soft-ICE does not use DOS or the ROM BIOS for its keyboard input or video output. You must have a video adapter compatible with MDA, Hercules, CGA, EGA, or VGA; however, the program can support two monitor configurations. Soft-ICE comes with a readable 100+ page manual and a 30-day, money-back guarantee.

Graphics/Windowing

Windows Software Development Kit
Microsoft Corporation
P.O. Box 97017
Bellevue, WA 98073
(800) 426-9400

Finally, Microsoft has released Version 2.0 of their Windows Software Development Kit (SDK) for DOS ($500), originally announced almost a year ago. The retail version of Windows/386 and Windows 2.0 has been available for several months. Both retail versions bear the subtitle "Presentation Manager"—clearly a misnomer that only extends the confusion in the muddy DOS and OS/2 waters. The Development Kit doesn't have the subtitle, so programmers shouldn't be confused about what they're getting. The SDK is designed for programmers developing applications to run under Windows 2.0 and Windows/386 under DOS. Several computer manufacturers (Compaq, Zenith, AT&T, etc.) have been bundling Windows/386 with their new machines.

The primary enhancements from Version 1.0x to 2.0 are overlapping windows, expanded memory support (LIM 4.0), dynamic data exchange, support for larger bitmaps, speed improvements, and reduced Windows kernel code size. The visual environment of Windows 2.0 is also supposed to "provide visual fidelity" (whatever that means) with the real OS/2 Presentation Manager when it finally arrives. The programming interface (system calls and particularly graphics routines) for Windows 2.0 is not the same under the OS/2 Presentation Manager even if the two might look outwardly similar. Time will tell how difficult it will be to port applications from one environment to the other.

The programmer's interface includes an additional 50 function calls and messages that emulate OS/2 program operations. New utilities include a SNAP utility (printing just portions of a screen), SPY (a message examiner), and a floating point library that senses and uses the 80x87 math chips. Most of the other utilities have been improved. Documentation has been significantly improved and consists of six manuals packaged in three binders. The Development Kit is distributed on both 5.25-inch (3 1.2 MB) and 3.5-inch (5 740K) disks (360K disks are available upon request). Upgrades from earlier 1.0x toolkit versions are $75, free if purchased since September 1987.

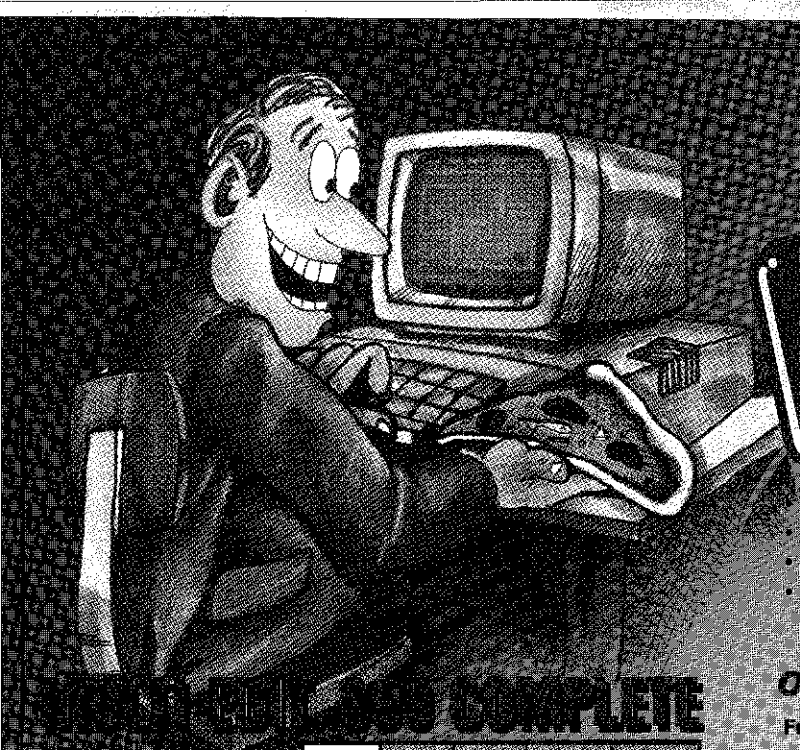
TMS 34010 Development Tools
Texas Instruments, Inc.
P.O. Box 1443
Houston, TX 77001
(800) 232-3200 (Ask for TMS34010.)

The TI TMS34010 and the Intel 82786 have been sparring for the high end PC and minicomputer market. The TI chip appears to be emerging with the larger market share. The TMS34010 Graphics System Processor is not just a graphics chip but is an advanced, high-performance 32-bit microprocessor optimized for graphics operations. Its general purpose programma-bility augmented with graphics operations implemented in hardware gives it many advantages over hardwired graphics controllers. The 34010 has 30 32-bit registers and instructions that can treat memory as a linear bitmap rather than a byte stream. The chip is currently available in 40- and 50MHz versions.

TI offers a series of hardware and software development tools for the 34010. Hardware for the PC is a TMS34010 Software Development Board (SDB) with a 34010 chip, a color palette chip, 256K of VRAM and 512K of program RAM, a serial UART, and a PC bus interface ($2495). The SDB supports a resolution of 640 x 480 pixels (1024 x 512 maximum) in 16 colors per scan line from a palette of 4096 and works with an analog RGB monitor (like NEC Multisync). However, similar boards supporting much larger screen resolutions, more colors, and on-board memory are now available from a variety of vendors. The hardware interface is a 4K block of PC memory at C7000h. The SDB comes with a PC Debugger and Development software to allow loading software on the SDB under DOS.

TI's Development software includes a variety of cross-development tools that run on either an IBM PC, TI PC, or a VAX (VMS, Berkeley, or System V Unix). The DOS tools include an Assembler package ($500), a C Compiler ($1000), and various support libraries. The Assembler package is a 34010 assembler (generates Unix COFF output), a linker, an archive utility, a ROM utility, and

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Multi-Edit vs. PIZZA

With EVERYTHING!

- Is your editor OUT TO LUNCH?
- Does it handle ALL OF YOUR NEEDS?
- Is it flexible, programmable and reconfigurable?
- MOST IMPORTANTLY, is it EASY TO USE?

OR WOULD YOU RATHER BE EATING PIZZA?

Only MULTI-EDIT tastes this good!

Fully automatic Windowing and Virtual Memory
Edit multiple files regardless of physical memory size
Easy cut-and-past between files
View different parts of the same file

Powerful, EASY-TO-READ high-level macro language
Standard language syntax
Full access to ALL Editor functions

Language-specific macros for C, PASCAL, BASIC and MODULA-2
Smart Indenting
Smart brace/parenthesis/block checking
Template editing
More languages on the way

Terrific word-processing features for all your documentation needs
Intelligent word-wrap
Automatic pagination
Full print formatting with justification, bold type, underlining and centering
Flexible line drawing
Even a table of contents generator

Compile within the editor
Automatically positions cursor at errors
Allocates all available memory to compiler

Complete DOS Shell
Scrollable directory listing
Copy, Delete and Load multiple files with one command
Background file printing

Regular expression search and translate
Condensed Mode display, for easy viewing of your program structure
Pop-up FULL-FUNCTION Programmer's Calculator and ASCII chart

and MOST IMPORTANT,
the BEST USER-INTERFACE ON THE MARKET!
Extensive context-sensitive help
Choice of full menu system or logical function key layout
Function keys are always labeled on screen (no guessing required!)
Keyboard may be easily reconfigured and re-labeled
Extensive mouse support
Easy, automatic recording and playback of keystrokes
Anchovies easily removed

...AND MORE! AND COMPLETE

	Multi-Edit	BRIEF 2.0	Norton Editor	Vedit Plus	PIZZA WITH EVERYTHING
Edit 20+ files at once using virtual memory	Yes	Yes	No	Yes	12 slices
Powerful high level macro language	Yes	Yes	No	Yes	Italian
Full UNDO	Yes	Yes	No	No	No
Visual marking of blocks	Yes	Yes	Yes	No	Looks good
Line, Stream and column Blocks	Yes	Yes	No	No	Use a knife
Automatic file save	Yes	Yes	No	No	No
Online help	Extensive	Limited	Limited	Limited	Extensive
Choice of keystroke commands or menu system	Yes	No	No	Yes	Menu Available
Function Key assignments labeled on screen (may be disabled)	Yes	No	No	No	No
Word processing functions	Extensive	Limited	Limited	Extra Cost	Difficult
Complete DOS shell	Yes	No	No	No	Deep Dish
Pop-up Programmer's Calculator and ASCII table	Yes	No	No	No ASCII	No
Unlimited Undo for Cut/Paste/Keystroke macros	Yes	No	No	Yes	Sauce on Cut/ others
Allocates all available memory to compiler when run from within Editor	Yes	No	No	No	Lots of bytes
Intelligent indenting, template editing and brace/parenthesis block matching and checking for C, PASCAL, BASIC and MODULA-2	Yes	C Only	No	Limited	Limited Intelligence
Flexible condensed mode display	Yes	No	Yes	No	Definitely
PRICE	**$99**	$195	$50	$195	About $17

Get Our FULLY FUNCTIONAL DEMO Copy for only $10!

To Order, Call 24 hours a day:
1-800-221-9280 Ext. 951
In Arizona: 1-602-968-1945
Credit Card and COD orders accepted.

American Cybernetics
1228 N. Stadem Dr.
Tempe, AZ 85281

Requires IBM/PC/XT/AT/PS2 or full compatible, 256K RAM, PC/MS-DOS 2.0 or later. Multi-Edit and American Cybernetics are trademarks of American Cybernetics. BRIEF is a trademark of Underware, Inc. Norton Editor is a trademark of Peter Norton Computing, Inc. Vedit is a registered trademark of CompuView Products Inc. Copyright 1987 by American Cybernetics.

MULTI-EDIT COMBINES POWER WITH EASE OF USE LIKE NO OTHER EDITOR ON THE MARKET TODAY.

Number 184 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Language and Compiler Design News

LALR, an LALR(1) parser generator, runs on MS-DOS computers and generates automata-based parsing engines for computer languages, command languages, and macro languages.

LALR reads a BNF grammar specification and outputs source code in Microsoft C, Turbo C, and UNIX C. It has successfully handled a COBOL grammar of 1561 productions, generating a parser with 2187 states in about 60 seconds.

LALR generated parsers are capable of processing the target language at a rate of approximately 14,000 lines per minute on an IBM PC/AT. Time includes scanning and parsing.

LALR provides an easy way to interface with a lexical scanner and the rest of your system. An advanced error recovery technique provides good repairs for syntax errors.

LALR comes with grammars for Ada, BASIC, C, and Pascal, source code for a lexical scanner, parser skeleton, test program, and calculator. It has a 60-day money-back guarantee. Price is **$99.**

LALR Research

1892 Burnt Mill / Tustin CA 92680

714-832-5257

Prepayment required, just $99. Same-day shipping.

Number 188 on the Reader Service Card

a 34010 simulator. The C Compiler comes with standard C library support. Additional libraries (licenses) include a Graphics/Math Function Library, a Bit-Map Font Library, a CCITT (FAX compression) Function library, and an IBM 8514/A Emulation library. TMS 34010 Technical manuals are also available.

HiREZ Mouse, EGA/Mouse
Logitech
6505 Kaiser Drive
Fremont, CA 94555
(800) 231-7717

If you need to do some mousing around, check out Logitech's several new products for the PC. Logitech's HiREZ mouse for the PC ($149) is designed expressly for the new generation of high resolution displays. The HiREZ mouse supports 320 dots per inch (dpi) resolution as compared with 200 or 100 for other mice and gives a full range of cursor travel with minimal movement. The HiREZ is a three button bus mouse that plugs into a half slot board. The package comes with mouse drivers, a mouse configuration utility, a menu building system, a mouse-based editor, and a software shell.

The Logitech Graphics System combines a bus mouse and an EGA (supports VGA 640 x 480 resolution in 16 colors) with a multi-sync-type monitor. The EGA and MOUSE share a single half-slot PC card, which may make it attractive for older IBM PC machines or other machines with a limited number of slots. The standard mouse is Logitech's 200 dpi resolution, but a HiREZ mouse can be used with the card. The EGA has 256K display memory, a light pen interface, and a feature connector and supports automatic software mode switching. The EGA board supports MDA, Hercules, and CGA, as well as EGA display adapters. A Windows driver is included. The 14-inch Autosync monitor supports a maximum resolution of 800 x 560. The EGA board and mouse retails for $399 and, bundled with the Autosync monitor, retails for $999. The EGA comes with a two-year, the mouse with a lifetime, warranty.

The *Logitech Mouse Programmer's Reference* ($5) includes a *Technical Reference Manual* (hardware), a *Mouse Programmer's Toolkit* (software reference to the Mouse Driver functions), and sample source code on disk. The software reference is 60+ pages and documents the INT 33h interface and shows example function calling sequences in assembler, Modula-2, C, Microsoft and Turbo Pascal, and the BASIC interpreter. The accompanying disk also includes a utility for designing custom mouse cursors.

DEMO II
Software Garden, Inc.
P.O. Box 373
Newton Highlands, MA 02161
(617) 332-2240

Dan Bricklin has released DEMO II ($195), a sequel to his highly successful DEMO program. DEMO II, like its predecessor, is a prototyping tool for designing interactive video demonstrations of a program. The original DEMO was powerful but supported text screens only. DEMO II extends support to graphics screens as well. Over 75 new features have been added. The major enhancements are bit-mapped graphic screen support, string and numeric variables while running, a wealth of new actions to execute while running (arithmetic operators, if, while, select/case, file, printer I/O, exec another program, etc.), and special effects when changing slides (paint from top to bottom, bottom to top, at various speeds, etc.).

Bit-mapped image support includes CGA 320 x 200 4-color, CGA 640 x 200 2-color, EGA 640 x 350 16-color, EGA 640 x 350 2-color, or Hercules 720 x 348 modes. The CAPTURE program supplied with DEMO II can capture text or bit-mapped graphics images from other programs. (PCX files from Zsoft PC Paintbrush—see Miscellaneous, *PJ*, 5.6—can be referenced.) Although DEMO II allows the incorporation of bit-mapped images, it does not support editing the images. Also, keep in mind that a 16 color EGA screen image may use 256K and that even compressed CAPTURE may limit the size of programs that run above it in memory to be captured from. DEMO II can use DOS environment strings for customization. The DBD DEMO file format has changed, but DEMO II can read older files. The program comes with a 220 page user's manual, keyboard function templates for both older style and newer keyboards, and an online tutorial. DEMO II comes with a license allowing an unlimited number of copies of the runtime (the original came with a license for up to 50 copies).

Parser Generators

LALR 3.0
LALR Research
1892 Burnt Mill
Tustin, CA 92680
(714) 832-LALR

LALR 3.0 ($99.00) is a fast, inexpensive LALR(1) parser generator tool for building translators or compilers. Parser generators can be used as tools in writing assemblers, macro processors, query processors, interpreters, compilers, and a variety of translators that take input in one form and translate it to output in some other. For example, you could use an LR parser to write a Turbo Pascal to C translator or to convert PostScript to HPGL. This software tool is similar to YACC (Yet Another Compiler-Compiler) under Unix. LALR (Look Ahead Left to right scanning, constructing a Rightmost derivation in reverse) parsers can be constructed for a wide class of grammars. The LR parsing method is one of the most general methods to parse a grammar, yet it can be very efficient. The main drawback is that it's too much work to construct an LR parser by hand. An LR parser usually consists of an input, an output, a stack, a driver program, and a parsing table.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The driver program is basically similar for all LR parsers, but the parsing tables would change for each parser constructed.

LALR3 reads an input specification for a grammar and generates scanner and LR parser (with parser tables) in C source code. The generated C code is targeted for Borland's Turbo C, Microsoft C, and Unix. By modifying the included parser skeleton code, you could write parsers in other languages. LALR3 can also provide a variety of listing details about the parser generated. The most difficult part of using one of these tools is writing the input grammar. The package includes several sample grammars for Ada, BASIC, C, and Pascal, a calculator, and LALR itself. The manual is about 65 pages covering the basics of operation but could go into greater detail on writing an input grammar. The LALR3 input grammar is different from the older YACC grammar and is similar to the grammar MetaWare used in their much more expensive Translator Writing System. A useful addition would be a translator to convert between a YACC input grammar and the LALR input format. In operation, LALR3 is fast and can swallow large grammars. Additional grammar files are available for FORTRAN-77, COBOL, and dBASE III at $99 each. The package comes with a 60-day, money-back guarantee.

QPARSER+ V3.0
QCAD Systems
1164 Hyde Avenue
San Jose, CA 95129
(800) 538-9787

QPARSER+ Translator Writing System ($300) is another tool for generating LR(1) parsers for compilers, translators, etc. But in addition to the LR parser and tables, QPARSER+ includes tools for syntax tree generation and symbol table management that might be used by a compiler. Translators can be generated in the source language of your choice. QPARSER+ reads an input grammar and produces parser tables that are combined with a skeleton parser driver file to generate a translator source file in Pascal or C. Three general-purpose skeleton parser files are provided in both Pascal and C. The C skeletons look more like translated Pascal code than like native C source code. One of the sample parser files supports abstract syntax tree generation.

Again, the most difficult part of using these tools is writing the input grammars. QCAD provides sample input grammars for a calculator, a P-code simulator, a Pascal subset compiler yielding P-code, and a tiny Pascal subset compiler yielding symbolic 8086 code. Like LALR's the QPARSER+ input grammar differs from the YACC grammar. The 200+ page manual describes use of the tools and discusses grammar definition and debugging. The latter part of the manual walks you through several of the sample grammar examples. An additional 200+ page spiral bound manual contains program listings. QPARSER+ runs comparably with LALR3 but can't swallow as large an input grammar. The program author Bill Barrett also co-authored the book *Compiler Construction: Theory and Practice* published by Science Research Associates (1986).

Assemblers/Tools

OPTASM
SLR Systems
1622 N. Main St.
Butler, PA 16001
(800) 833-3061

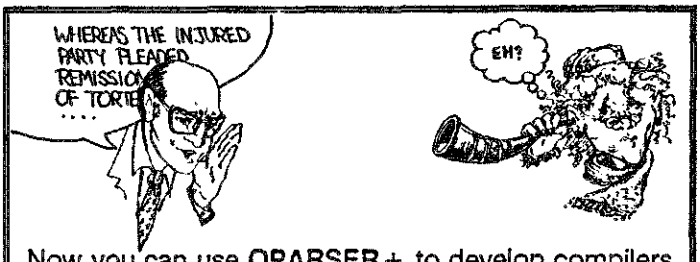
Need a shot of speed for your 8086 assembly language development? SLR systems has released OPTASM ($195), a fast macro assembler for the PC compatible with Microsoft's MASM. SLR Systems has a track record for writing fast, powerful assemblers. Several years back, SLR developed the fastest Z80 and 8080 assemblers that operated under CPM/80 (Z80 machines). Written completely in optimized Z80 assembly language, these tools sped by and replaced the existing competition from Microsoft, Digital Research, and others. At long last, they've brought out an 8086 as-

QUICK BASIC ↓ C

$199/$375

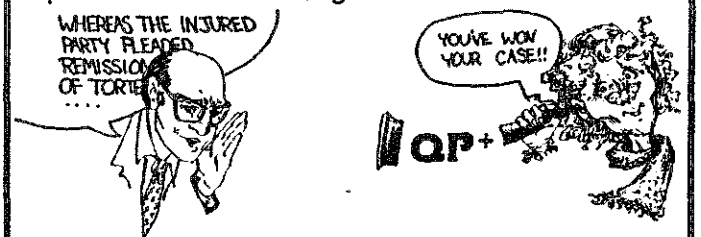
translator
GOTOLESS CONVERSION
BOX 835910, Richardson
TX 75083, 214-404-1405

Number 127 on the Reader Service Card



Now you can use QPARSER+ to develop compilers, interpreters, complex user-interfaces, language & file format translators (i.e. Pascal to C, Bit Map to Postscript), language debuggers like lint, etc.

Develop language translators in C and Pascal within the IBM PC/XT/AT or VAX/VMS environments. A new user manual, automated syntax tree construction and an advanced code generation language are just a few of the improvements over the original QPARSER.



Another translation by QPARSER+™
Limited Time Offer Just $300 — FREE demo disk available
QCAD Systems
1164 Hyde Avenue, San José CA 95129 (408) 727-6884
Outside Calif. call TOLL-FREE (800) 538-9787

Number 189 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

sembler. Like its Z80 predecessor, OPTASM was written in assembly language for speed and was designed for compatibility with MASM, the defacto standard on the PC, including even its quirks. OPTASM supports almost all MASM features and constructs (Versions 3.0, 4.0, and 5.0) except generating 80386 opcodes (.386 and .387 directives and DF farwords) and the /ZI option for CodeView. I'm betting they'll eventually be added.

While MASM remains a two-pass assembler, OPTASM is an N-pass assembler so that forward references are automatically resolved and jumps can be optimized, meaning no more phase errors. OPTASM also has a number of features and options not available with MASM. A configuration utility lets you customize the default OPTASM options. Options may be set via the default configuration, an environment variable, command-line options, or source code directives in that order of precedence. OPTASM features include local procedure labels and making all symbols public along with other useful MASM extensions. I tested both OPTASM and MASM 5.0 on an old version of Kermit (v2.26) from Columbia University (about 378K of source in 12 files). OPTASM benchmarked at about three times the speed of MASM while creating 58K of OBJ files and over 1 megabyte of listings. The version I tested in March had only a preliminary manual, so I won't comment on its omissions until I receive a final copy. The preliminary manual did cover configuration, options and features, and basic syntax.

C Programmers Toolbox
MMC AD Systems
Box 360845
Milpitas, CA 95035
(408) 263-0781

MMC AD Systems offers two C Programmers Toolboxes (Volumes I and II) with a variety of software development tools for C programmers. Volume I (Version 1.2—$79.95) contains 12 programs, some rather mundane but a few very powerful. The simpler tools include programs equivalent to Unix utilities under different names: Cat (copy to stdout), CharCnt (file byte count), FileDump (binary file dump), Tail, FileComp (binary file compare), ExecTime (time the execution of a program), and a TransLit program for substituting or deleting characters from an input stream. If these were the only tools, this package would be hardly worth mentioning, but Volume I also contains three primary tools: CPrint (a C pretty printer), PMon/MapVar (a pair of programs to extract symbol information from a link map file and monitor/profile the execution of a program), and CFlow (a C program function hierarchy analyzer). Volume I comes with a comprehensive 140 page user's manual.

If you port software, CFlow makes this entire package a bargain, taking either a list of files with .C extension (wildcards allowed) on the command line or from an input file and generating several detailed user selectable reports. These include a raw flow report of defined and referenced functions, an alphabetical list of functions with functions that call them and functions referenced, a hierarchical tree or structured flow diagram, and an undefined function list. I found the tree hierarchy and undefined (referenced but not defined) subroutine lists most valuable. The undefined list provides a list of library routines to check against your favorite compiler's library. CFlow swallowed with ease and quickly 3 megabytes of source code for the GNU C Compiler along with large subdirectories of X Windows C source code. The reports identified which Unix BSD routines I would need to write or emulate and the basic structure of these larger programs.

Volume II (Version 1.0—$79.95) contains several minor tools: Detab (replace tabs with spaces), Entab (replace spaces with tabs), FileDiff (text file comparison), Filelist (list/locate files from multiple directories), Fill (argument merge text utility), and Strip (extract a section of text from a file). The three primary tools are CXref (a multiple file cross reference generator), CLint (additional syntax checking of multi-module C source code), and CritPath (determines critical paths of a program). Whereas CFlow handles functions, CXref handles both functions and variables, identifying unused variables and potential variable conflicts. Volume II has a 200+ page user's manual.

Both toolboxes support a similar command line interface (wildcards and input response files) and send output to stdout (which may be redirected). The tools I tried operated both quickly and flawlessly. You can buy Volumes I and II together for $130. The toolboxes come with a 30-day, money-back guarantee, which you're unlikely to use.

Books

Programmer's Guide to PC and PC/2 Video Systems
by Richard Wilton
Microsoft Press
(800) 638-3030

The Programmer's Guide to PC and PC/2 Video Systems ($24.95) is a solid source book on common video display systems for the IBM PC and PS/2 computers (544 pages). Richard Wilton stays true to his title, covering the video adapters made by IBM and Hercules. These include the MDA, CGA, EGA, MCGA (PS/2 Models 25 & 30), and VGA. Hercules coverage includes the basic Hercules Graphics Card (HGC), the HGC Plus (Ramfont), and the Hercules InColor Card. Wilton uses a hands-on approach with examples and routines in assembly language and C. He provides both the hardware and software overview of PC video, then focuses in depth on programming using both the ROM BIOS interface and direct hardware programming (direct screen writes, CRTC chip programming, etc.). The book covers programming in both and text and graphics modes, exploiting RAM-based alphanumeric character sets, pixel programming, line, circle, ellipse, and arc drawing primitives, region filling, and animation. Later chapters describe more advanced programming techniques using interrupts, bit blocks, bit-plane layering, panning on the EGA and VGA, and the light pen interface. One of the appendices provides code to identify video subsystems. I found this a very solid resource book and fault it only for not covering or at least mentioning several of the more common video adapters such as the AT&T 640 x 400 monochrome graphics adapter (also used by Compaq and Toshiba), and the enhanced EGA and VGA boards available from many vendors. A companion disk that contains about 6,000 lines of code is available ($21.95).

Advanced C Programming for Displays
by Marc Rochkind
Prentice-Hall Software Series
Prentice-Hall
Englewood Cliffs, NJ 07632
(201) 592-2000

Marc Rochkind has written another fine book, this time on programming for text displays in a variety of environments ($32.95, 331 pages). Rochkind restricts his focus to video and window programming using text modes, but covers text display under both DOS and several Unix environments. He focuses on character-based displays rather than graphics for several reasons: Character-based programs are simpler (clearer examples), they use less memory and can update the display faster (performance), and practically all displays can show characters (universality). The book is designed around layers of abstraction, beginning with abstract objects (characters and attributes) followed by the physical screen, the physical keyboard, the virtual keyboard, windows, and virtual screens. To illustrate techniques, Rochkind builds on a simple editor that gets enhanced in successive chapters. A real strength of this book is its clear coverage of Unix Curses (both old and new) and the Unix Termcap database along with DOS (both ROM BIOS and memory mapped video programming). The programming examples have been designed to work in the same way under both Unix and DOS. Programming is in C with an appendix of assembly language routines for DOS BIOS and screen interfacing. Rochkind provides some interesting test data comparing Curses and Termcap. The programs were tested in several environments—DOS, Xenix V (IBM AT), Berkeley Unix (VAX), and Unix System V (AT&T 3B2). A companion disk available from the author contains the 9,000 lines of code printed in the book. By the way, the earlier Marc Rochkind book—one I refer to regularly—is *Advanced Unix Programming* (1985). That book contains excellent material on interprocess communication. ♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LOW COST ADVERTISING

If you sell your products or services to professional programmers, then a **PJ Billboard** ad is an easy, low cost way to reach more potential buyers.

PJ Billboard ads must be prepaid and are non-commissionable. Call the ad department for prices.

Programmer's Journal
P.O. Box 30160
Eugene, OR 97403
503-747-0800

DRUMA FORTH-83 SYSTEM LANGUAGE

Strict **'83 STANDARD**. Developed for in house use. No 64K limitation. 64K speed & compactness to 320K.

* **IBM PC/XT/AT** & compatibles, **DOS 2.0+**
* **ROM**: Headerless code, separated variables
* Editor, assembler, file & DOS interfaces
* Many powerful and innovative features
* Full Pkg. **$79**, Demo **$10**, S&H **$2**, **VISA/MC**

Druma Forth-83, Druma Inc.
P.O. Box 610097
Austin, TX 78761
512-323-0403 Orders
512-323-2402 BBoard **RS# 172**

TURBO PASCAL PROGRAMMERS

The SOURCE CODE for TURBO Pascal is now available. For a remarkably low price users can now have fully commented source code for TURBO Pascal v3.01A (binary arithmetic). This includes the run-time library, editor and compiler. Everything you need is included, even an assembler! Free information.

Price: IBM-PC $67.50, Z-80 $45 (specify disk required)

C.C. Software
1907 Alvarado Ave. #B
Walnut Creek, CA 94596
415-939-8153 **RS# 132**

FST MODULA-2

Fast two pass compiler with integrated editor and make, large memory model, inline assembler. Requires PC with 512K minimum. Support via BBS and BIX ('fst' conference).

Evaluation copy $12, registered version (includes library source code) $39.

PMI's Repertoire® available.

Fitted Software Tools
P.O. Box 956129
Duluth, GA 30136
BBS 404-497-0931 **RS# 138**

ADVANCED SOURCE CODE ANALYSIS

Advanced C Source Code Analysis with C: LINES/C: TREE

* Outlined listings reflect flow of control within procedures.
* Tree diagrammer shows flow of control between procedures.
* Source code reformatter

$59.95, shipping & handling included. Maryland residents add 5% sales tax. Requires PC Compatible, MS-DOS, 256K.

SoftRex
4807 Bethesda Ave., #287
Bethesda, MD 20814
301-881-8274 **RS# 185**

C-TRIEVE ISAM LIBRARIES

Compatible with X/OPEN ISAM standard.

* FAST!
* No run-time royalties.
* Part of a family of SQL data base products (inquiries invited).
* Supports transactions and journaling
* Contains extensions to X/OPEN standard
* Full telephone support provided
* Excellent documentation
* Attractively priced

Coromandel Industries Inc.
P.O. Box 620
Gracie Station
New York, NY 10028
212-772-6489 **RS# 192**

PJ CODE IS AVAILABLE!

On disk from:

Programmer's Journal
503-747-0800
See ad on page 72.

On BBS:

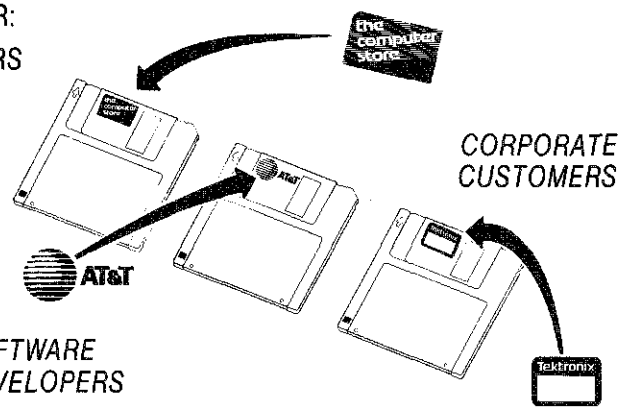
FIDO/OPUS Network
Blazer Net 152 Node 9
503-343-9724

Mnematics Videotex
P. O. Box 19
Sparkhill, NY 10976-0019
800-322-3633
914-761-4700 in NY

Turbo Source Search
P.O. Box 876
Scituate, MA 02066
617-545-9131

"NEW" PRINTED 3.5" SHUTTERS

FOR:
VARS



CORPORATE CUSTOMERS

SOFTWARE DEVELOPERS

Permanent Identification of your Company, Information, or Products.

SOFTWARE DUPLICATION/PACKAGING

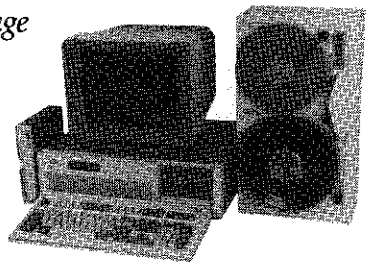
3.5" / 5.25" / HIGH DENSITY

WEST COAST TELECOM
CALL: (503) 620-1888

Number **190** on the Reader Service Card

9-Track Tape Subsystem

The Data Interchange Solution You've Been Waiting For!



Qualstar's ½" 9-track Ministreamers™ bring full ANSI data interchange capability to your PC or Macintosh™ computer system.

With 9-track tape, you are free to exchange data files with any mainframe or minicomputer in the world.

Our affordable Ministreamers come in both 7" and 10½" versions and use less desk space than an ordinary sheet of paper. They can provide 1600 thru 6250 BPI capability and may be used for disk backup as well as data interchange. Complete subsystem prices, including software, start as low as $2,495 for 7" units and at $3,670 for 10½" units.

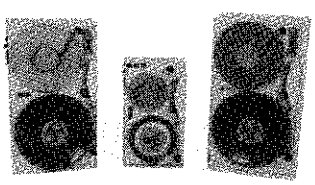
Qualstar has become the market leader in desk-top 9-track systems for a good reason; our tape drives have established an outstanding record for reliability and low cost of ownership.

Discover the many advantages 9-track tape has over other Micro/Mainframe links.

Call us today!

QUALSTAR®

9621 Irondale Avenue,
Chatsworth, CA 91311
Telephone: (818) 882-5822



Macintosh is a trademark of Apple Computer, Inc.

Number **97** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ADVERTISER INDEX

Reader Service#	Advertiser Name	Page Number
180	A.I. Architects, Inc.	49
184	American Cybernetics	75
191	Andsor Research, Inc.	33
162	Anji Software, Inc.	32
170	ASCII	66
86	Aspen Systems	43
30	Austin Code Works	10
51	Az-Tech Software, Inc.	43
187	Az-Tech Software, Inc.	66
2	Blaise Computing	3
3	Borland International	C4
132	C. C. Software	79
38	C Ware Corporation	23
56	CompuView Products, Inc.	1
192	Coromandel Industries, Inc.	79
*	Creative Programming Consultants	23
124	Custom Real-Time Software	24
5	Dair Computer Systems	19
172	Druma, Inc.	79
70	E.C.I.	15
8	Ecosoft, Inc.	26
173	Elan Computer Group, Inc.	37
113	Elfsoft	50
138	Fitted Software Tools	79
118	Fleming Software	15
135	Gilmore Systems	42
*	Gimpel Software	28
127	Gotoless Conversion	77
122	Hauppauge Computer Works, Inc.	51
154	HavenTree Software LTD	36
169	IGC	5
155	Komputerwerk	71
63	Laboratory Microsystems, Inc.	19
44	Lahey Computer Systems	67
188	LALR Research	76
141	Merlin Publishing Group	50
143	Metagraphics Software Corporation	22
45	MetaWare, Inc.	55
*	PLEASE CONTACT THESE ADVERTISERS DIRECTLY	

Reader Service#	Advertiser Name	Page Number
168	Michael J. Young Software Engineering	19
179	Mightysoft, Inc.	35
121	MISOSYS Inc.	67
94	MMC AD Systems	34
109	NanoSoft	63
160	Nu-mega Technologies	53
16	Opt-Tech Data Processing	15
102	Oregon Software	71
145	Pathfinder Associates	25
*	Programmer's Journal Mail Order	72
6	Periscope Company, The	11
146	Programmer's Connection	7
189	QCAD Systems	77
97	Qualstar Corporation	79
147	Quarterdeck Office Systems	C2
174	Seidl Computer Engineering	65
148	SemWare	62
161	Serengeti Software	58
175	Sharpe Systems Corporation	54
186	Silver State Software	23
165	SLR Systems	6
33	Soft Advances	29
177	Soft Evolutions, Inc.	36, 48
185	SoftRex	79
98	Software Family, The	27
136	Sophco, Inc.	73
158	Summit Information Systems, Inc.	8
85	Sunny Hill Software	37
74	Sydetech, Inc.	29
171	Systemat Corporation	18
166	Tool Makers, The	67
25	TurboPower Software	64
35	Turbo Source Search	58
125	V-Communications	29
190	West Coast Telecom, Inc.	79
73	Whitewater Group, The	8
123	Xenosoft	48
182	Zortech, Inc.	C3
*	PLEASE CONTACT THESE ADVERTISERS DIRECTLY	

When contacting any of the above advertisers, please tell them you saw their ad in ***Programmer's Journal*** and that you appreciate their continuing support of this magazine.

WRITE FOR

Programmer's Journal

Share your hot technical tips, cool coding ideas or hard earned industry wisdom with your fellow programmers. Send your article to

Editor
Programmer's Journal
P.O. Box 30160
Eugene, OR 97403

ADVERTISE IN

Programmer's Journal
July/August Issue, Volume 6.4
Space Reservation Deadline:
May 6, 1988
Materials Due:
May 20, 1988
Call ***Programmer's Journal*** Ad Department to reserve your space today!

(503) 747-0800

PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TOOLKITS FOR TURBO C & QUICK C from ZORTECH INC.

HOTKEY

A complete set of Terminate Stay Resident (TSR) functions that help you to write reliable 'pop-up' programs.

Now you can make your programs 'Sidekickable'. Two example programs are included, a 'pop-up Calculator' and a pop-up 'Critical Error Handler'.

The Hotkey toolkit handles all floating point functions in resident mode.

The 32 page manual includes an interesting discussion of the origin and history of undocumented MS-DOS function calls, together with a full explanation of the theory and practical use of TSR's.

Only $49.95! (State Turbo C or Quick C version.)

COMMS

Do you need to incorporate serial communications into your applications? Yes! Then get this inexpensive but highly professional COMMS toolkit from Zortech Inc.

Look at the list of features: Xmodem, Kermit and ASCII file transfer, Hayes modem control, VT52, VT100 and ANSI terminal emulation, supports up to 8 serial ports, speeds up to 19.2k baud rate and higher.

Two demonstration programs are included, MINICOM and MAXICOM (like Procomm) together with the 120 page manual and full source code FREE!

Only $49.95! (State Turbo C or Quick C version.)

GAMES

Have you ever wondered how to write a chess program? Now we reveal the secret algorithms and techniques of the masters with this dynamic Games toolkit.

The package comes complete with the full source code to three ready to play games of strategy – Chess, Backgammon and Wari (an ancient African game).

A comprehensive 150 page manual is provided giving an in depth look at the history, structure and program design of such 'Strategy Games'.

Only $49.95!

(State Turbo C or Quick C version.)

SUPERTEXT

This is not simply an 'Editor' toolkit, but a full-blown, 'WordStar' compatible wordprocessor with the full source code.

As well as all the normal editing functions, you will also find 'dot' commands and full printer control. The SuperText toolkit handles files of any size and allows full on-screen configuration.

Do you need to incorporate a wordprocessor into your application? Yes! Then get the SuperText toolkit complete with full source code and 150 page manual now!

Only $49.95! (State Turbo C or Quick C version.)

PROSCREEN

Generate high quality data entry screens with the Pro-Screen – Screen Designer and Code Generator.

You can draw the data entry screen, define the input fields, define the input criteria, set screen colors and attributes, draw single or double lines, make boxes – press a few buttons and 'hey presto' Pro-Screen generates the C source code for your application!

Professional applications programmers will find this versatile utility and it's associated functions invaluable.

Comes complete with a substantial 78 page manual and demo programs.

Only $49.95! (State Turbo C or Quick C version.)

WINDOWS

Add super-fast text screen handling to your applications with the WINDOWS library from Zortech Inc.

Give your applications the professional look – with instant zooming and exploding windows. Incorporate drop-down menus and Lotus style menus with our easy to use functions.

Automatically handles memory saving and buffering of window text. Use any number of overlapping windows in your applications. Write to any window, read from any window, close any window, pull any window to the top.

Over 55 functions together with a big 85 page manual and remember, you get the full source code.

Only $49.95! (State Turbo C or Quick C version.)

NEW! C VIDEO

- *Now learn C the easy way!*
- *Get the 'Complete C Video Course' from Zortech Inc. together with our big 365 page workbook.*
- *Ten 1 hour tapes – 36 lessons!*
- *Easy to follow course, you get an excellent introduction to the C language.*
- *Takes you step-by-step up to the intermediate and advanced levels.*
- *Teach yourself at home or the office – at your own speed.*

only ~~$295.00~~ **$199.95**

Yes! Rush me these items!

☐ HOTKEY
☐ COMMS
☐ PRO-SCREEN
☐ WINDOWS ☐ GAMES
☐ SUPERTEXT ☐ C VIDEO

FREE SHIPPING – VISA/MC/COD/CHECK

Name .....

Address .....

..... **Phone** .....

VISA or MC # ..... **Exp. Date** .....

ZORTECH Inc. 361 Massachusetts Ave, Arlington, MA 02174
Orders & Enquiries Tel: (617) 646 6703

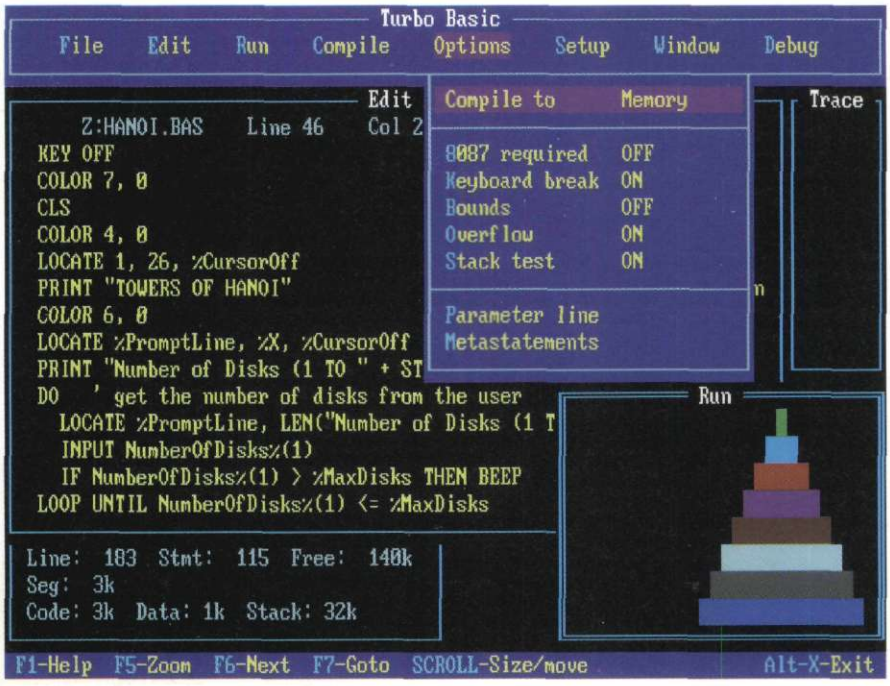
ORDER HOTLINE (800) 848-8408

Reader Service Number 70

ZORTECH
BOSTON · LONDON · FRANKFURT · GENEVA

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Basically speaking, there's one choice . . . Turbo Basic!



Turbo Basic's development environment gives you overlapping windows, pull down menus, and the ability to run text-based applications in a window.

Turbo Basic® is the BASIC that lets even beginners write polished, professional programs almost as easily as they can write their names.

The others don't. When you really examine them, you'll find that even though they may be "quick," they make it hard to get where you're going. (Sort of like a car with an engine but no steering wheel.)

Turbo Basic takes you farther faster—in the comfort of a sleek development environment that gives you full control. Naturally it has a slick, fast compiler just like all Borland's technically superior Turbo languages. It *also* has a full-screen windowed editor, pull-down menus, and a trace debugging system. And innovative Borland features like binary disk files, true recursion, and more control over your compiling. Plus the ability to create programs as large as your system's memory can hold.

The critics agree. The choice is basic. Turbo Basic from Borland.

"... What really makes Turbo Basic special is its blinding speed, small size, and many added commands. Programs compiled with Turbo Basic are often much faster and smaller than those produced by other compilers.

Ethan Winer, PC Magazine Best of 1987

Turbo Basic, simply put, is an incredibly good product.

William Zachman, Computerworld"

Add another Basic advantage: The Turbo Basic Toolboxes New!

- ***The Database Toolbox*** gives you code to incorporate into your own programs. You don't have to reinvent the wheel every time you write new Turbo Basic database programs. New!
- ***The Editor Toolbox*** is all you need to build your own text editor or word processor, including source code for two sample editors.

*60-Day Money-back Guarantee**

Compare the BASIC differences!

	Turbo Basic 1.1	QuickBASIC 4.0 Compiler	QuickBASIC 4.0 Interpreter
Compile & Link to stand-alone EXE	*3 sec.*	7 sec.	——
Size of .EXE	*28387*	25980	——
Execution time w/80287	*0.16 sec.*	16.5 sec.	21.5 sec.
Execution time w/o 80287	*0.16 sec.*	286.3 sec.	292.3 sec.

The Elkins Optimization Benchmark program from March 1988 issue of Computer Language was used. The Program was run on an IBM PS/2 Model 60 with 80287. The benchmark tests compiler's ability to optimize loop-invariant code, unused code, expression and conditional evaluation.

System Requirements: For the IBM PS/2™ and the IBM® family of personal computers and all 100% compatibles. Operating System: PC-DOS (MS-DOS) 2.0 or later. Toolboxes require Turbo Basic 1.1. Memory: 384K RAM for compiler, 640K RAM to compile Toolboxes.

*Customer satisfaction is our main concern; if within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.

All Borland products are trademarks or registered trademarks of Borland International, Inc. Other brand and product names are trademarks of their respective holders. Copyright ©1988 Borland International, Inc. BI 1246

BORLAND INTERNATIONAL

For the dealer nearest you call (800) 543-7543

Number 3 on theReader Service Card

PJ 5/6 88

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED